REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
11/10/2011	Final	21 June 2010 -- 20 June 2011

4. TITLE AND SUBTITLE
Emergent Intelligent Behavior through Integrated Investigation of Embodied Natural Language, Reasoning, Learning, Computer Vision, and Robotic Manipulation.

5a. CONTRACT NUMBER

5b. GRANT NUMBER
N00173-10-1-G023

5c. PROGRAM ELEMENT NUMBER
55-1005-10

6. AUTHOR(S)
Siskind, Jeffrey, M.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Purdue University
School of Electrical and Computer Engineering
465 Northwestern Avenue
West Lafayette, IN 47907-2035

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
4555 Overlook Ave. SW
Washington, DC 20375

10. SPONSOR/MONITOR'S ACRONYM(S)
NRL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release, distribution is unlmited

13. SUPPLEMENTARY NOTES
20111019240

14. ABSTRACT
We developed methods for estimating the camera-relative pose and compositional structure of a Lincoln Log assembly from a single image together with methods for rendering a description of that structure in English and then replicating that structure with a robot manipulator arm from that English description.

15. SUBJECT TERMS
robotics, computer vision, natural language processing, AI, machine learning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU		

104686

REPORT OF INVENTIONS AND SUBCONTRACTS

(Pursuant to "Patent Rights" Contract Clause) (See Instructions on Reverse Side.)

Form Approved
OMB No. 9000-0095
Expires Aug 31, 2001

1.a. NAME OF CONTRACTOR / SUBCONTRACTOR	c. CONTRACT NUMBER	2.a. NAME OF GOVERNMENT PRIME CONTRACTOR	c. CONTRACT NUMBER	3. TYPE OF REPORT *(x one)*
Purdue University	N00173-10-1-G023			a. INTERIM [ ] X b. FINAL [X]
b. ADDRESS *(Include ZIP Code)*	d. AWARD DATE *(YYYYMMDD)*	b. ADDRESS *(Include ZIP Code)*	d. AWARD DATE *(YYYYMMDD)*	4. REPORTING PERIOD *(YYYYMMDD)*
610 Purdue Mall, Room 300 West Lafayette, IN 47907-2040	20100621			a. FROM 20100621 b. TO 20110620

SECTION I - SUBJECT INVENTIONS

5. "SUBJECT INVENTIONS" REQUIRED TO BE REPORTED BY CONTRACTOR/SUBCONTRACTOR *(If "None," so state)*

NAME(S) OF INVENTOR(S) *(Last, First, Middle Initial)* a.	TITLE OF INVENTION(S) b.	DISCLOSURE NUMBER, PATENT APPLICATION SERIAL NUMBER OR PATENT NUMBER c.	ELECTION TO FILE PATENT APPLICATIONS (X) d. (1) United States (a) Yes	(b) No	(2) Foreign (a) Yes	(b) No	CONFIRMATORY INSTRUMENT OR ASSIGNMENT FORWARDED TO CONTRACTING OFFICER (x) e. (a) Yes	(b) No
	NONE							

f. EMPLOYER OF INVENTOR(S) NOT EMPLOYED BY CONTRACTOR/SUBCONTRACTOR		g. ELECTED FOREIGN COUNTRIES IN WHICH A PATENT APPLICATION WILL BE FILED	
(1)(a) Name of Inventor *(Last, First, Middle Initial)*	(2)(a) Name of Inventor *(Last, First, Middle Initial)*	(1) Title of Invention	(2) Foreign Countries of Patent Application
(b) Name of Employer	(b) Name of Employer		
(c) Address of Employer *(Include ZIP Code)*	(c) Address of Employer *(Include ZIP Code)*		

SECTION II - SUBCONTRACTS *(Containing a "Patent Rights" clause)*

6. SUBCONTRACTS AWARDED BY CONTRACTOR/SUBCONTRACTOR *(If "None," so state)*

NAME OF SUBCONTRACTOR(S) a.	ADDRESS *(Include ZIP Code)* b.	SUBCONTRACT NUMBER(S) c.	FAR "PATENT RIGHTS" d. (1) Clause Number	(2) Date (YYYYMM)	DESCRIPTION OF WORK TO BE PERFORMED UNDER SUBCONTRACT(S) e.	SUBCONTRACT DATES *(YYYYMMDD)* f. (1) Award	(2) Estimated Completion

SECTION III - CERTIFICATION

7. CERTIFICATION OF REPORT BY CONTRACTOR/SUBCONTRACTOR *(Not required if: (X as appropriate))* [ ] *Small Business or* [ ] *Non-Profit organization. (X appropriate box)*

I certify that the reporting party has procedures for prompt identification and timely disclosure of "Subject Inventions," that such procedures have een followed and that all "Subject" Inventions" have been reported.

a. NAME OF AUTHORIZED CONTRACTOR/SUBCONTRACTOR Official (Last, First, Middle Initial)	b. TITLE Special Assistant to the Vice Provost for Research	c. SIGNATURE	d. DATE SIGNED
Millsaps, Mary D.	Operations Manager, Office of the Vice President for Research		7/29/2011

DD Form 882, Jan 1999 (I

PREVIOUS EDITION MAY BE USED

WHS/DIOR, Jan 99

FEDERAL FINANCIAL REPORT

(Follow form instructions)

1. Federal Agency and Organizational Element to Which Report is Submitted	2. Federal Grant or Other Identifying Number Assigned by Federal Agency (To report multiple grants, use FFR Attachment)	Page	of
Naval Research Laboratory	N00173-10-1-G023	1	1 pages

3. Recipient Organization (Name and complete address including Zip code)

Purdue University
Sponsored Program Services
155 S. Grant Street
West Lafayette, IN. 47907 - 2114

4a. DUNS Number	4b. EIN	5. Recipient Account Number or Identifying Number (To report multiple grants, use FFR Attachment)	6. Report Type	7. Basis of Accounting
07-205-1394	35-6002041	104686	☒ Quarterly ☐ Semi-Annual ☐ Annual ☒ Final	☐ Cash ☒ Accrual

8. Project/Grant Period From: (Month, Day, Year)	To: (Month, Day, Year)	9. Reporting Period End Date (Month, Day, Year)
6/21/2010	6/20/2011	6/20/2011

10. Transactions	Cumulative
(Use lines a-c for single or multiple grant reporting)	
Federal Cash (To report multiple grants, also use FFR Attachment):	
a. Cash Receipts	39,120.28
b. Cash Disbursements	49,949.63
c. Cash on Hand (line a minus b)	**(10,829.35)**
(Use lines d-o for single grant reporting)	
Federal Expenditures and Unobligated Balance:	
d. Total Federal funds authorized	50,000.00
e. Federal share of expenditures	49,949.63
f. Federal share of unliquidated obligations	-
g. Total Federal share (sum of lines e and f)	**49,949.63**
h. Unobligated balance of Federal funds (line d minus g)	**50.37**
Recipient Share:	
i. Total recipient share required	-
j. Recipient share of expenditures	-
k. Remaining recipient share to be provided (line i minus j)	-
Program Income:	
l. Total Federal program income earned	-
m. Program income expended in accordance with the deduction alternative	-
n. Program income expended in accordance with the addition alternative	-
o. Unexpended program income (line l minus line m or line n)	-

11. Indirect Expense	a. Type	b. Rate	c. Period From	Period To	d. Base	e. Amount Charged	f. Federal Share
	Predetermined	54.0%	6/21/2010	6/20/2011	30,494.15	16,466.92	16,466.92
				g. Totals:	**30,494.15**	**16,466.92**	**16,466.92**

12. *Remarks: Attach any explanations deemed necessary or information required by Federal sponsoring agency in compliance with governing legislation:*

13. **Certification: By signing this report, I certify that it is true, complete, and accurate to the best of my knowledge. I am aware that any false, fictitious, or fraudulent information may subject me to criminal, civil, or administrative penalties. (U.S. Code, Title 18, Section 1001)**

a. Typed or Printed Name and Title of Authorized Certifying Official	c. Telephone (Area code, number and extension)
Helen Moschinger, Senior Account Manager	**765-494-1078**
	d. Email address: helenm@purdue.edu
b. Signature of Authorized Certifying Official: Hmoschinger	e. Date Report Submitted (Month, Day, Year): 8/26/11
	14. Agency use only:

Standard Form 425
OMB Approval Number: 0348-0061
Expiration Date: 10/31/2011

Paperwork Burden Statement
According to the Paperwork Reduction Act, as amended, no persons are required to respond to a collection of information unless it displays a valid OMB Control Number. The valid OMB control number for this information collection is 0348-0061. Public reporting burden for this collection of information is estimated to average 1.5 hours per response, including time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding the burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to the Office of Management and Budget, Paperwork Reduction Project (0348-0060), Washington, DC 20503.

Final Report

Emergent Intelligent Behavior through Integrated Investigation of Embodied Natural Language, Reasoning, Learning, Computer Vision, and Robotic Manipulation

Purdue University
Jeffrey Mark Siskind
N00173-10-1-G023
21 June 2010–20 June 2011
Monday 10 October 2011

1 Efforts under this grant

Our efforts under this grant have focused on the following activities:

- We developed a method for determining the camera-relative pose of an assembly constructed out of LINCOLN LOGS from a single image from an uncalibrated camera. This method uses the inherent structure of the LINCOLN LOG assembly domain which has orthogonally oriented logs on planar surfaces parallel to the ground plane resulting in two predominant directions of image edges when projected to the image plane.
- We developed a method for determining the structure (component logs and their 3D placement) of a LINCOLN LOG assembly from a single image from an uncalibrated camera. This method uses a novel visual language model that encodes the inherent structure of the LINCOLN LOG assembly domain and uses this language model to improve detection accuracy over what is possible from raw low-level detector output.
- We extended the above structure-estimation method to support:
  - Determination of confidence in ones structure estimate.
  - Determination of what is occluded and what is visible.
  - Improving the reliability of ones structure estimate by integrating information from multiple views, to observe occluded parts of the structure.
  - Improving the reliability of ones structure estimate by disassembling parts of the structure to render occluded parts visible.
- We developed a method for generating English descriptions of the structure of a LINCOLN LOG assembly as recovered from visual input.
- We developed a method for controlling a robot manipulator arm to replicate a LINCOLN LOG assembly from an English description.
- We developed a method to allow two robot manipulator arms to collaborate when building a LINCOLN LOG assembly. One robot serves as an *inventory manager* inventorying the parts available on its work surface. The second robot serves as the *builder*, requesting parts from the inventory manager and placing them in the assembly under construction.

All of the above are fully documented in the papers that we have published or submitted for publication, along with the web sites we have prepared, as outlined below.

2 Deliverables

Our statement of work states:

> Under this grant we will investigate and conduct research on methods for determining the pose and structure of LINCOLN LOG assemblies from one or more images of such assemblies taken from one or more viewpoints at one or more stages of partial disassembly together with optional linguistic specification of constraints on the observed assembly. We will use those methods to drive robotic disassembly of LINCOLN LOG assemblies whose structure has been so determined. And we will quantitatively evaluate the performance of these methods. The results of this research effort will be detailed in a final report to be delivered on or by 31 October 2010.
>
> We will deliver a final report, on or by 31 October 2010, detailing the methods developed along with a quantitative evaluation of their performance.

Award of the grant was delayed so that the initial period of performance was extended until 20 November 2011. A no-cost extension extended the period of performance through 20 June 2011. We have accomplished the objectives outlined in the above statement of work and published our results (or in some cases submitted our results for publication) as outlined below. We attach the published, accepted, and submitted papers as appendices to this report.

3 Publications

3.1 Published

Siddharth, N., Barbu, A., and Siskind, J.M., A Visual Language Model for Estimating Object Pose and Structure in a Generative Visual Domain, ICRA, 2011.

3.2 Accepted

Wingate, D., Goodman, N.D., Stuhlmueller, A., and Siskind, J.M., Nonstandard Interpretations of Probabilistic Programs for Efficient Inference, NIPS, 2011.

3.3 In Review

Siddharth, N., Barbu, A., and Siskind, J.M., Seeing Unseeability to See the Unseeable, submitted to ICRA 2012.

3.4 Invited Presentations

Siskind, J.M., Mediating Cross-Modal Perception, Motor Control, Language, and Reasoning with Common and Deep Semantic Representations, AAAI workshop on Language-Action Tools for Cognitive Artificial Agents: Integrating Vision, Action, and Language, 2011

3.5 Web Sites

The following web sites acknowledge funding under this contract:
`https://engineering.purdue.edu/~qobi/mindseye/`
(Click on *Online appendix for the kickoff meeting* to see the acknowledgment. The README files included in many of the downloadable releases also contain the acknowledgment.)

`http://upplysingaoflun.ecn.purdue.edu/~qobi/cccp/`
(Click on *Research* and followed by either *Our Robotic Testbed*, *Structure Estimation from a Single View*, *Structure Estimation from Multiple Views*, *Structure Estimation from Partial Disassembly*, *Structure Assembly and Disassembly*, or *Robot Collaboration* to see the acknowledgment.)

`https://engineering.purdue.edu/~qobi/icra2011/`

`https://engineering.purdue.edu/~qobi/icra2012/`

4 Execution of funding

Our cumulative expenses for 21 June 2010–20 June 2011 total $49,949.63.

A Visual Language Model for Estimating Object Pose and Structure in a Generative Visual Domain

Siddharth Narayanaswamy, Andrei Barbu, and Jeffrey Mark Siskind

Abstract— **We present a generative domain of visual objects by analogy to the generative nature of human language. Just as small inventories of phonemes and words combine in a grammatical fashion to yield myriad valid words and utterances, a small inventory of physical parts combine in a grammatical fashion to yield myriad valid assemblies. We apply the notion of a language model from speech recognition to this visual domain to similarly improve the performance of the recognition process over what would be possible by only applying recognizers to the components. Unlike the context-free models for human language, our visual language models are context sensitive and formulated as stochastic constraint-satisfaction problems. And unlike the situation for human language where all components are observable, our methods deal with occlusion, successfully recovering object structure despite unobservable components. We demonstrate our system with an integrated robotic system for disassembling structures that performs whole-scene reconstruction consistent with a language model in the presence of noisy feature detectors.**

I. Introduction

Human language is *generative*:[1] a small inventory of phonemes combine to yield a large set of words and then this inventory of words combine to yield a larger set of utterances. Systems that process language must deal with the combinatorial nature of generativity. The probability of correct word recognition becomes fleetingly small with even a slight probability for error in phoneme recognition and the probability of determining the correct parse of an utterance becomes fleeting small with even a slight probability for error in word recognition. This is remedied with a *language model*, a specification of which combinations of phonemes constitute valid words and which combinations of words constitute valid utterances. Such a language model often takes the form of a *grammar*.

The vast majority of computer-vision research in pose estimation and object recognition deals with nongenerative collections of objects. Such nongenerative collections require distinct models or exemplars for each object (class) that varies greatly in shape, structure, or appearance. We instead present an approach for doing pose estimation and structure recognition in generative visual domains, analogous to the approach for human language. We illustrate this approach with the domain of LINCOLN LOG assemblies. LINCOLN LOGS is a children's assembly toy with a small component inventory. We limit this inventory to three component types: 1-notch, 2-notch, and 3-notch logs. These combine in myriad ways to yield a large set of assemblies. We present low-level feature detectors that collect evidence for the components in a fashion analogous to low-level feature detectors in speech recognizers. But as in speech, the probability of correct recognition of an entire assembly becomes fleetingly small with even a slight probability for error in log recognition. We remedy this with a *visual language model* or a *grammar* of LINCOLN LOGS, a specification of which combinations of logs constitute valid assemblies.

The analogy breaks down in two ways requiring novel methods. First, most computer models of speech and language assume that the grammar is context free. This allows a top-down tree-structured generative process where the generation of siblings is independent. In contrast, the symbolic structure underlying LINCOLN LOG assemblies takes the form of graphs with cycles and thus the visual language model is context sensitive and is formulated as a stochastic constraint-satisfaction problem. Second, in language, all of the components are observable; at least in principle, one can obtain perceptual evidence of each phoneme in a word and each word in an utterance. In contrast, visual domains exhibit *occlusion*; it is almost always necessary to determine object structure without perceptual evidence for all of the components. Our methods address both of these issues. Our work builds upon the notion that scenes and objects are represented as descriptions involving parts and spatial relations [1]–[10], differing from prior work in the extreme degree of generativity of the LINCOLN LOG domain. None of this prior work focuses on domains that can generate as large a class of distinct structures from as small a class of components. Moreover, we focus on determining the precise pose and structure of an assembly, including the 3D pose of each component, with sufficient accuracy to support robotic manipulation and, in particular, the ability to robotically construct a symbolically precise replicate of a structure from a single image.

LINCOLN LOG structures are composed out of a small inventory of components, namely 1-notch, 2-notch, and 3-notch logs. As shown in Fig. 1, such logs are characterized by a small number of shape parameters: the inter-notch distance l_1, the log diameter l_2, and the distance l_3 from a log end to the closest notch center. Valid structures contain logs arranged so that their notches are aligned and their medial axes are parallel to the work surface. Thus valid structures

http://engineering.purdue.edu/~qobi/icra2011

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907, USA {snarayan,abarbu,qobi}@purdue.edu

[1]We mean the Chomskyan sense of generative, not the sense in contrast to discriminative. Indeed, while our domain is generative in the Chomskyan sense, our recognizer uses a discriminative model.

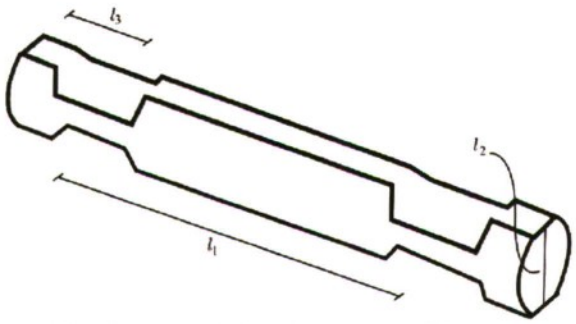


Fig. 1. The 3D geometric shape parameters of LINCOLN LOGS.

have logs on alternating layers j at height $l_2(j+0.5)$ oriented along one of two orthogonal sets of parallel lines spaced equally with horizontal distance l_1. The lines for even layers are mutually parallel, the lines for odd layers are mutually parallel, and the projections of a line from an even layer and an odd layer onto the work surface are perpendicular. We refer to this set of lines as the *grid* (see Fig. 2). This grid imposes a symbolic structure on the LINCOLN LOG assembly. Symbolic grid coordinates (i,j,k) map to metric camera-relative coordinates (x,y,z) by the parameters l_1, l_2, and l_3 together with the structure pose: the transformation from the grid coordinate system to the camera coordinate system. Estimating the structure of a LINCOLN LOG assembly thus reduces to two phases: estimating the structure pose (section II) and determining the log occupancy at each symbolic grid position (section III).

II. ESTIMATING THE STRUCTURE POSE

Before beginning these two phases, we first compute a mask that separates the LINCOLN LOG structure in the image foreground from the background. We manually collect 20–30 image segments of LINCOLN LOG components and compute the mean μ and covariance Σ of the pixel values in these segments in a five-dimensional color space UVHSI. We then derive a mask M from an input image I containing those pixels p with values whose Mahalanobis distance from μ is less than or equal to a threshold t:

$$M_p = \begin{cases} 1 & \|C(I_p) - \mu\|_\Sigma \leq t \\ 0 & \text{otherwise} \end{cases}$$

where C denotes the map from input pixel values to UVHSI.

Nominally, the structure pose contains six degrees of freedom corresponding to translation and rotation about each axis. To simplify, we assume that the structure rests on the horizontal work surface. Thus we fix vertical translation, roll around the camera axis, and pitch around the horizontal axis perpendicular to the camera axis to be zero, leaving only three free parameters: horizontal translation of the structure along the work surface and yaw around the vertical axis. To resolve the periodic translation ambiguity in the symbolic grid coordinate system, we assume that the minimum occupied i, j, and k values are zero. We further assume that we know the symbolic grid size: the maximum occupied i, j, and k values.

Images of LINCOLN LOG assemblies contain a predominance of straight edges that result from log edges. Given this, we estimate the structure pose in a two-step process. We first find the pose p that maximizes the coincidence between the set $L(p)$ of projected grid lines l_g and the set L_I of image-edge line segments l_i:

$$\underset{p}{\operatorname{argmin}} \sum_{l_i \in L_I, l_g \in L(p)} \|l_i, l_g\|$$

where $\|l_i, l_g\|$ denotes the Euclidean distance between the midpoint of a line segment and its closest point on a line, weighted by the disparity in orientation between the line and the line-segment. We then refine this pose estimate by maximizing the coincidence between projected grid lines and the set P_I of image edge points p_i:

$$\underset{p}{\operatorname{argmin}} \min_{p_i \in P_I, l_g \in L(p)} \|p_i, l_g\|$$

where $\|p_i, l_g\|$ denotes the Euclidean distance between a point and the closest point on the line. We use a soft min function [11]–[13] when computing the latter with gradient-based methods (reverse-mode automatic differentiation [14]).

To obtain L_I, we apply a Canny edge detector [15] together with the KHOROS line finder [16] to extract linear edge segments from the input image, discarding short segments and those that do not lie wholly within the mask region defined by M. We then select the edge segments corresponding to the two most prominent edge orientations, by placing the segments into bins according to their orientation and selecting the edge segments in the two largest bins. To obtain P_I, we apply Phase Congruency [17] to the input image I to compute the orientation image $O(I)$. Each pixel in $O(I)$ contains a quantized orientation. We chose P_I to be those pixels whose quantized orientation is closest to the mean edge-segment orientations of the above two largest bins.

This two-step process offers several advantages. The first step converges quickly but exhibits error in the recovered prominent edge orientations. The second step estimates pose more accurately (typically within 5mm translation and 2° rotation), but only with close initial estimates, such as those provided by the first step.

Fig. 2 illustrates successful pose estimation of several LINCOLN LOG structures. Note that we estimate the pose of a target object from a single image without any knowledge of the specific 3D shape or structure of that object, without any prior training images of that object in different poses, using only generic information from the domain, namely that the object is a valid LINCOLN LOG assembly.

III. DETERMINING THE LOG OCCUPANCY AT EACH SYMBOLIC GRID POSITION

The symbolic grid positions $q = (i,j,k)$ refer to points along log medial axes at notch centers. Each such grid position may be either unoccupied, denoted by $\emptyset$, or occupied with the n^{th} notch, counting from zero, of a log with m notches, denoted by (m,n). For each grid position we wish to

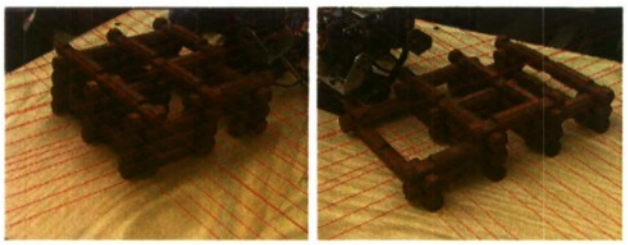


Fig. 2. Estimating the pose of an arbitrary LINCOLN LOG assembly and the symbolic grid thus imposed on the assembly.

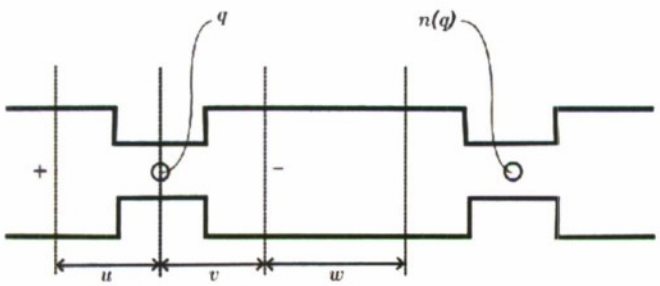


Fig. 3. The random variables Z_q^+ and Z_q^- that correspond to log ends for grid position q and the random variables Z_q^u, Z_q^v, and Z_q^w that correspond to log segments.

determine its occupancy, one of seven possibilities: $\emptyset$, $(1,0)$, $(2,0)$, $(2,1)$, $(3,0)$, $(3,1)$, and $(3,2)$. We construct a discrete random variable Z_q for each grid position q that ranges over these seven possibilities.

We determine several forms of image evidence for the log occupancy of a given grid position. LINCOLN LOGS, being cylindrical structures, generate two predominant image features: ellipses that result from the perspective projection of circular log ends and line segments that result from the perspective projection of cylindrical walls. We refer to the former as *log ends* and the latter as *log segments*. Log ends can potentially appear only at distance $\pm l_3$ from grid positions along the direction for the layer of that grid position. We construct boolean random variables Z_q^+ and Z_q^- to encode the presence or absence of a log end at such positions. There are two kinds of log segments: ones corresponding to l_1 and ones corresponding to l_3. Given this, we construct three boolean random variables Z_q^u, Z_q^v, and Z_q^w for each grid position q that encode the presence or absence of log segments for the bottoms of logs, i.e., log segments between a grid position and the adjacent grid position below. Z_q^u and Z_q^v encode the presence or absence of a log segment of length l_3 behind and ahead of q respectively, along the direction for the layer of q while Z_q^w encodes the presence or absence of a log segment of length $l_1 - 2l_3$ between grid positions along the same layer. Fig. 3 depicts the log ends and log segments that correspond to a given grid position as described above.

We formulate a stochastic constraint-satisfaction problem (CSP [18]) over these random variables. The constraints encode the validity of an assembly. We refer to these constraints as the *grammar* of LINCOLN LOGS (section III-C). We take image evidence to impose priors on the variables Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w (sections III-A and III-B) and solve this stochastic CSP to perform structure estimation (section III-D).

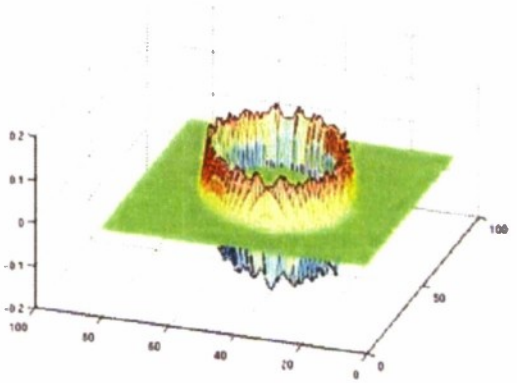


Fig. 4. Elliptical edge filter for detecting log ends

A. *Evidence for the presence or absence of logs*

Given the pose p, a log end present as the result of Z_q^+ or Z_q^- being true will manifest as an ellipse of known shape, size, and position in the image. We use $x^+(p,q)$, $y^+(p,q)$, $a^+(p,q)$, $b^+(p,q)$, and $\theta^+(p,q)$ to denote the parameters (center, lengths of major and minor axes, and orientation of major axis) of an ellipse that would manifest from Z_q^+ and similarly for Z_q^-. We find these parameters by a least-squares fit of 20 equally spaced 3D points on the log end projected to the image. The 3D points can be determined in closed form from the grid position q and the parameters l_1, l_2, and l_3. We then construct an indicator function $f(x,y)$ with the value 1 for points (x,y) inside the ellipse and the value 0 for points outside the ellipse and convolve this with a Laplacian of a Gaussian filter, $\text{LoG}(r,\sigma)$, to obtain an elliptical edge filter $E(x,y,a,b,\theta)$ (Fig. 4). Nominally, a high response to this filter applied to an image correlates with the presence of an elliptical feature with parameters x, y, a, b, and θ. To provide robustness in the face of inaccurate pose estimation, we compute the maximal filter response in a 5-dimensional region centered on x, y, a, b, and θ derived by perturbing each axis a small amount.

Similarly, given the pose p, a log segment present as the result of Z_q^u, Z_q^v, or Z_q^w being true will manifest as a line segment between known image points. We denote the points for Z_q^u as $(x_1^u(p,q), y_1^u(p,q))$ and $(x_2^u(p,q), y_2^u(p,q))$ and similarly for Z_q^v and Z_q^w. These image points can be determined in closed form by projecting the 3D points derived from the pose p, the grid position q, and the parameters l_1, l_2, and l_3.

In principle, we could use a similar filter method to determine evidence for log segments. However, log ends usually yield highly pronounced edges because logs are never stacked horizontally end to end. Log are often stacked vertically and the log segments between two such vertically stacked logs would yield less-pronounced edges. Thus we use a more sensitive method to determine evidence for log segments. Given the pose p of the structure, we recompute the prominent edge orientations o_1 and o_2 using the methods from section II (this time applied to the output of the

second step of pose estimation, not the first, to give a more accurate estimate of these orientations). For each prominent orientation o, we compute the disparity between o and $O(I)$ at each pixel, compute the prominence at each pixel by attenuating the disparity, and scale the energy image, $E(I)$, by this prominence: $W(I,o) = E(I) \circ \cos^2(O(I) - o)$. This constitutes a graded edge map for edges with orientation o. We search a rectangular region in $W(I,o)$, after thresholding, for the longest line segment. The search region corresponds to a dilation of the rectangle bounded by the endpoints of the target log segment. The length of the longest line segment found correlates with the presence of the target log segment.

B. Mapping evidence to priors

We train a mapping function from evidence to priors for the log-segment and log-end evidence functions respectively on a set of 30 images annotated with ground truth, i.e., true positives and true negatives, along with occlusion. For each evidence function, we bin their respective raw, real-valued responses into 20 bins and annotate each bin with the percentage of responses that are true positives and the central response value for that bin. The annotated bins correspond to a discrete sequence of impulses with impulse magnitude representing the percentage of true positives for the central response value. We then employ a weighted linear interpolation function between impulses to provide the mapping function. The weighting factor e typically takes the form of a real value $e \in (0,1)$.

C. The grammar of Lincoln Logs

We refer to the adjacent grid position below q as $b(q)$, the adjacent grid position further from the origin along the direction of the grid lines for the layer of q as $n(q)$, and the adjacent grid position closer to the origin along the direction of the grid lines for the layer of q as $p(q)$. Ignoring boundary conditions at the perimeter of the grid, the grammar of LINCOLN LOGS can be formulated as the following constraints:

a) 2-notch logs occupy two adjacent grid points

$$Z_q = (2,0) \leftrightarrow Z_{n(q)} = (2,1)$$

b) 3-notch logs occupy three adjacent grid points

$$\begin{aligned} Z_q = (3,0) &\leftrightarrow Z_{n(q)} = (3,1) \\ Z_q = (3,0) &\leftrightarrow Z_{n(n(q))} = (3,2) \\ Z_{n(q)} = (3,1) &\leftrightarrow Z_{n(n(q))} = (3,2) \end{aligned}$$

c) 1- and 2-notch logs must be supported at all notches

$$Z_q \in \{(1,0),(2,0),(2,1)\} \rightarrow Z_{b(q)} \neq \emptyset$$

d) 3-notch logs must be supported in at least 2 notches

$$Z_q = (3,0) \rightarrow \begin{pmatrix} (Z_{b(q)} \neq \emptyset \wedge Z_{b(n(q))} \neq \emptyset) \vee \\ (Z_{b(q)} \neq \emptyset \wedge Z_{b(n(n(q)))} \neq \emptyset) \vee \\ (Z_{b(n(q))} \neq \emptyset \wedge Z_{b(n(n(q)))} \neq \emptyset) \end{pmatrix}$$

e) log ends must be at the ends of logs

$$\begin{aligned} Z_q^- &\leftrightarrow Z_q \in \{(1,0),(2,0),(3,0)\} \\ Z_q^+ &\leftrightarrow Z_q \in \{(1,0),(2,1),(3,2)\} \end{aligned}$$

f) short log segments indicate occupancy above or below

$$\begin{aligned} Z_q^u &\leftrightarrow (Z_q \neq \emptyset \vee Z_{b(b(q))} \neq \emptyset) \\ Z_q^v &\leftrightarrow (Z_q \neq \emptyset \vee Z_{b(b(q))} \neq \emptyset) \end{aligned}$$

g) long log segments indicate presence of a multi-notch log above or below

$$Z_q^w \leftrightarrow \begin{pmatrix} \begin{pmatrix} Z_q \in \{(2,0),(3,0),(3,1)\} \wedge \\ Z_{n(q)} \in \{(2,1),(3,1),(3,2)\} \end{pmatrix} \vee \\ \begin{pmatrix} Z_{b(b(q))} \in \{(2,0),(3,0),(3,1)\} \wedge \\ Z_{b(b(n(q)))} \in \{(2,1),(3,1),(3,2)\} \end{pmatrix} \end{pmatrix}$$

To handle the boundary conditions, we stipulate that the grid positions beyond the perimeter are unoccupied, enforce the support requirement (constraints c–d) only at layers above the lowest layer, and enforce log-segment constraints (f–g) for the layer above the top of the structure.

D. Structure estimation

To perform structure estimation we first establish priors over the random variables Z_q^+ and Z_q^- that correspond to log ends and the random variables Z_q^u, Z_q^v, and Z_q^w that correspond to log segments using image evidence and establish a uniform prior over the random variables Z_q. This induces a probability distribution over the joint support of these random variables. We then marginalize the random variables that correspond to log ends and log segments and condition this marginal distribution on the language model Φ. Finally, we compute the assignment to the random variables Z_q that maximizes this conditional marginal probability.

$$\underset{\mathbf{Z}}{\operatorname{argmax}} \sum_{\substack{\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w \\ \Phi[\mathbf{Z},\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w]}} \Pr\left(\bigwedge_q Z_q, Z_q^+, Z_q^-, Z_q^u, Z_q^v, Z_q^w\right)$$

To speed up the conditional marginalization process, we prune assignments to the random variables that violate the grammar Φ using arc consistency [19]. To speed up the maximization process, we use a branch-and-bound algorithm [20] that maintains upper and lower bounds on the maximal conditional marginal probability. Without both of these, structure estimation would be intractable.

An alternate method to perform structure optimization is to establish the same priors over the random variables that correspond to log ends and log segments but parametrize the priors over the random variables Z_q. We then marginalize over all random variables, computing this marginal probability over the parameterized priors for the random variables Z_q. We then search over this parameter space for the distributions over the random variables Z_q that maximize this marginal probability. We do this using the reduced-gradient optimization algorithm [21], [22] where the gradients are calculated using reverse-mode AD. The linear constraints are used to constrain the parameters of the probability distribution to be nonnegative and sum to one. Ideally, we'd prefer to use the latter method exclusively, but the former method is faster to compute for the relatively larger assemblies when compared to the latter.

E. Occlusion

Nominally, with the above method, one derives evidence for the presence or absence of log ends and log segments of the various kinds at every possible grid position. In other words, one uses image evidence to impose a prior on all of the random variables Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w. However, some of these log ends and log segments may be occluded. If we *know* that a log end or log segment is occluded then we ignore all evidence for it from the image, giving it chance probability of being occupied. With this, the grammar can often fill in the correct values of occluded random variables for both log ends and log segments, and thus determine the correct value for an occluded Z_q. The question then arises: how does one determine whether a log end or log segment is occluded? We propose the following method. One first assumes that all of the log ends and log segments on the frontal faces of the grid are visible but all other log ends and log segments are occluded. One then performs structure estimation under this initial hypothesis. With the recovered structure estimate, one determines log-end and log-segment visibility by projective geometry given the known pose, and iterates this process until convergence. We have recently implemented this algorithm and expect to report on its performance in the future. All experiments reported in section IV were performed with manual annotation of occlusion information. Note that we only annotate for a given symbolic log-segment or log-end *position* whether or not it is *visible*, **not** whether or not that position is *occupied* with a log segment or log end. The latter is determined automatically.

IV. Experimental results

We took images of 32 distinct LINCOLN LOG structures, each from 5 distinct poses resulting in a total of 160 images. We performed foreground-background separation and pose estimation for all 160 images using the methods from section II. Pose was estimated within 5mm translation and 2° rotation of ground truth for 142 images. We discarded the 18 images with inaccurate pose estimation and performed structure estimation on the remainder. The results for 5 images, all of distinct structures, are shown in Fig. 7. Fig. 7(a) was derived by thresholding the priors on Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w at $t = 0.5$. Fig. 7(b–d) were derived by solving a stochastic CSP with various subsets of the constraints and rendering the values of Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w for the solution provided by the first method in section III-D. Fig. 7(e) was derived by solving the stochastic CSP with all constraints and rendering the values of Z_q for the solution provided by the first method in section III-D. Note that our method determines the correct component type (Z_q) of most occluded logs in the assemblies in the second row of Fig. 7(e). It gives an incorrect component type for only a single log in that row.

We conducted experiments to determine how much the grammar improves the accuracy of structure estimation. We performed variants of the runs in Fig. 7(a-d), varying the threshold t and the mapping from evidence to priors to produce the ROC curves depicted in Fig. 5. The mapping

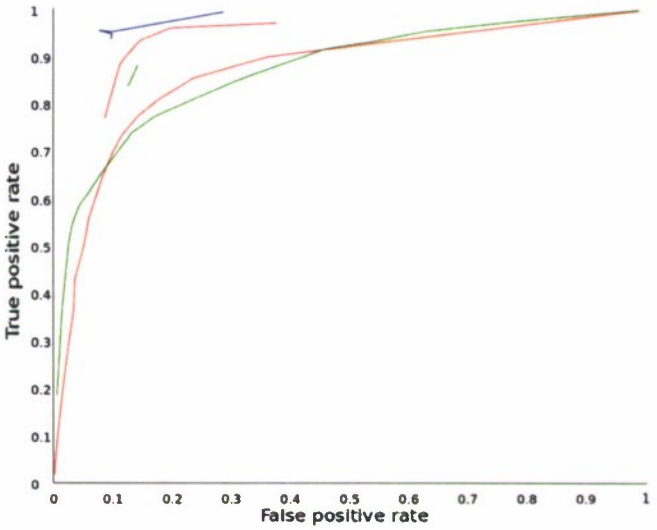


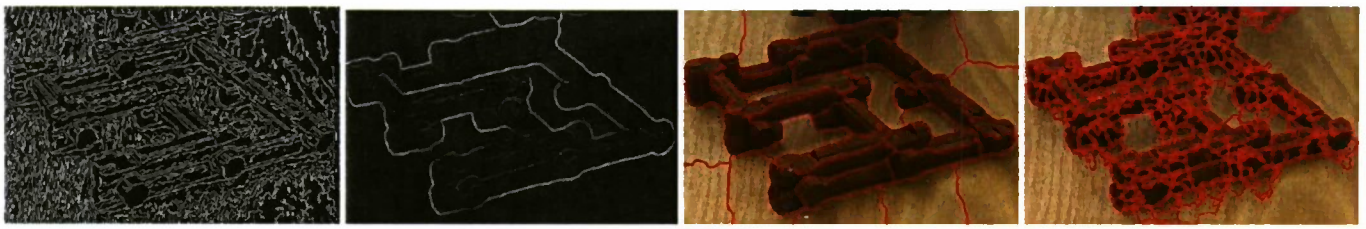
Fig. 5. ROC curves. The lower green and red curves constitute the ROC for the log-end and log-segment detectors respectively with varying thresholds t without the grammar. The upper green curve measures ROC for Z_q^+ and Z_q^- under constraints a–e varying the mapping from evidence to priors. The upper red curve measures ROC for Z_q^u, Z_q^v, and Z_q^w under constraints a–d and f–g varying the mapping from evidence to priors. The blue curve measures ROC for Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w under all constraints varying the mapping from evidence to priors.

function is varied through the weighting factor e for the linear interpolator discussed in section III-B.

Pose and structure estimation is sufficiently robust to support robotic manipulation. Supplementary material included on the website for this paper contains videos of fully autonomous robotic disassembly of six different LINCOLN LOG structures whose pose and structure have been determined from a single image as well as videos of semiautonomous robotic assembly of replicate LINCOLN LOG structures from the same estimated pose and structure.

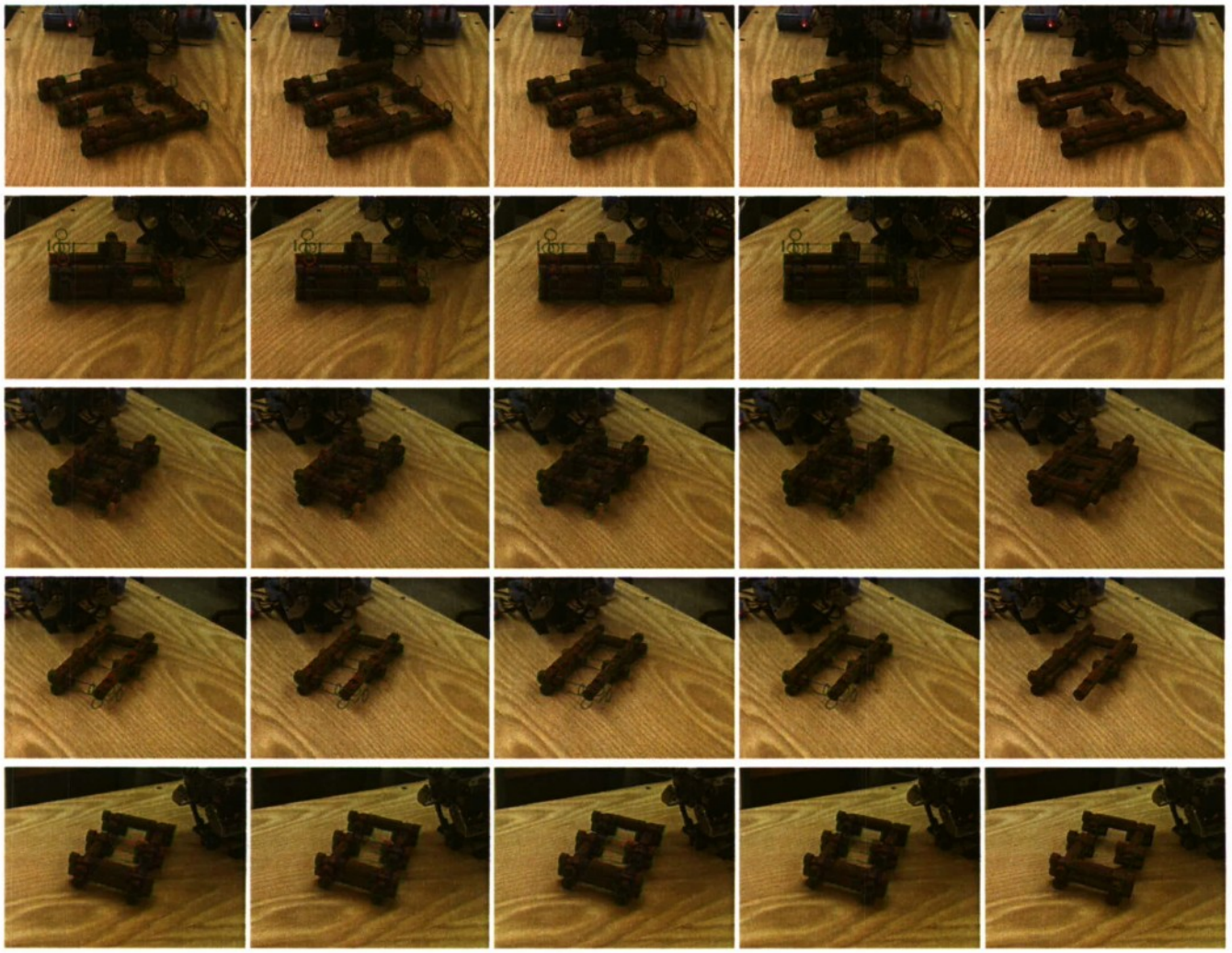
V. Conclusion

LINCOLN LOGS are children's toys yet the computational problem we present *is not* a toy. Pose and structure estimation of LINCOLN LOG assemblies is *far* more difficult than may appear on the surface. The space of objects to be recognized is combinatorially large. Much of every structure is in self occlusion. The low contrast due to shadows and color, intensity, and texture uniformity make it impossible to recognize even *visible* logs with existing techniques. No standard edge detector (e.g., Canny [15] or PB [23]) can reliably find edges separating adjacent logs or circular log ends and no standard segmentation method (e.g., Normalized Cut [24] or Mean Shift [25]) can reliably find log parts *even when fully visible* as shown in Fig. 6. Even our filter-based feature detectors, which use pose information along with constraints from the language model to *tune to the expected feature at the expected image position*, produce correct binary decisions only about 65% of the time. Occlusion only makes matters worse. Performing non-stochastic constraint satisfaction (e.g., Waltz line labeling [26]) on the binary



(a) (b) (c) (d)

Fig. 6. A comparison with a number of standard edge detectors and segmentation methods. Neither (a) MATLAB's Canny edge detector nor (b) the PB edge detector reliably find edges separating adjacent logs or log ends. Neither (c) Normalized Cut nor (d) Mean Shift segment out the log parts.



(a) (b) (c) (d) (e)

Fig. 7. (a) Raw detector response. (b) Detector response with just constraints a–d and f–g. (c) Detector response with just constraints a–e. (d) Detector response with all constraints. (e) Estimated structure. In (a–d), bright red indicates true negative, dark red indicates false negative, bright green indicates true positive, dark green indicates false positive, and blue indicates occlusion. In (e), green indicates true positive and red indicates false negative. There are no false positives and true negatives are not indicated. We suggest that the reader view this figure at a high magnification level in a PDF viewer to appreciate the images.

output of these detectors leads to inconsistent CSPs on *all* images in our dataset.

We have demonstrated a visual domain that is generative in much the same way that human language is generative. We have presented a visual language model that improves recognition accuracy in this domain in much the same way that language models improve speech-recognition accuracy. Unlike context-free models of human language, our visual language models are context sensitive and formulated as stochastic CSPs. Much of our visual experience in the artifactual world is perceiving generative man-made structures like buildings, furniture, vehicles, etc. Our LINCOLN LOG domain is a first step towards building visual language models for such real-world domains.

Language models for vision are more complex than those for human language as they must deal with occlusion resulting from perspective projection and pose variation. However, visual domains exhibit a novel possibility: recovering structure despite occlusion by integrating the perceptual evidence from multiple images of the same object taken from different poses. In the LINCOLN LOG domain, one can carry this even further. When faced with ambiguity arising from occlusion, a robot can partially disassemble a structure to view occluded substructure and integrate perceptual evidence from multiple images taken at different disassembly stages to yield a complete unambiguous estimate of the structure of the original assembly prior to disassembly. Moreover, it is possible to integrate information about pose or structure from different modalities. One can integrate partial pose and structure information from one or more images with partial pose and structure information expressed in human language to yield a complete unambiguous estimate of pose and structure. We are, in fact, able to do this and expect to report on this in the future.

VI. ACKNOWLEDGMENTS

This work was supported, in part, by NSF grant CCF-0438806, by the Naval Research Laboratory under Contract Number N00173-10-1-G023, by the Army Research Laboratory accomplished under Cooperative Agreement Number W911NF-10-2-0060, and by computational resources provided by Information Technology at Purdue through its Rosen Center for Advanced Computing. Any views, opinions, findings, conclusions, or recommendations contained or expressed in this document or material are those of the author(s) and do not necessarily reflect or represent the views or official policies, either expressed or implied, of NSF, the Naval Research Laboratory, the Office of Naval Research, the Army Research Laboratory, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

REFERENCES

[1] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 200, no. 1140, pp. 269–94, 1978.

[2] I. Biederman, "Recognition-by-components: A theory of human image understanding." *Psychological review*, vol. 94, no. 2, pp. 115–47, 1987.

[3] W. Wang, I. Pollak, T.-S. Wong, C. A. Bouman, M. P. Harper, and J. M.Siskind, "Hierarchical stochastic image grammars for classification and segmentation," *IEEE Trans. on Image Processing*, vol. 15, pp. 3033–52, 2006.

[4] S.-C. Zhu and D. Mumford, "A stochastic grammar of images," *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 4, pp. 259–362, 2006.

[5] J. M. Siskind, J. S. Jr., I. Pollak, M. P. Harper, and C. A. Bouman, "Spatial random tree grammars for modeling hierarchal structure in images with regions of arbitrary shape," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1504–19, 2007.

[6] L. L. Zhu, Y. Chen, and A. Yuille, "Unsupervised learning of a probabilistic grammar for object detection and parsing," in *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

[7] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *Proceedings of the 10th European Conference on Computer Vision*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 30–43.

[8] M. Aycinena Lippow, L. P. Kaelbling, and T. Lozano-Perez, "Learning grammatical models for object recognition," in *Logic and Probability for Scene Interpretation*, ser. Dagstuhl Seminar Proceedings, no. 08091, Dagstuhl, Germany, 2008.

[9] V. Savova and J. Tenenbaum, "A grammar-based approach to visual category learning," in *Proceedings of the 30th Annual Meeting of the Cognitive Science Society*, 2008.

[10] V. Savova, F. Jäkel, and J. Tenenbaum, "Grammar-based object representations in a scene parsing task," in *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*, 2009.

[11] S. Smale, "Algorithms for solving equations," in *Proceedings of the International Congress of Mathematicians*, 1986, pp. 172–95.

[12] B. Chen and P. T. Harker, "A non-interior-point continuation method for linear complementarity problems," *SIAM Journal of Matrix Analysis Applications*, vol. 14, no. 4, pp. 1168–90, 1993.

[13] C. Kanzow, "Some noninterior continuation methods for linear complementarity problems," *SIAM Journal of Matrix Analysis Applications*, vol. 17, no. 4, pp. 851–68, 1996.

[14] B. Speelpenning, "Compiling fast partial derivatives of functions given by algorithms," Ph.D. dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, Jan. 1980.

[15] J. Canny, "A computational approach to edge detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–98, 1986.

[16] K. Konstantinides and J. R. Rasure, "The Khoros software development environment for image and signal processing," *IEEE Trans. on Image Processing*, vol. 3, pp. 243–52, 1994.

[17] P. Kovesi, "Image features from phase congruency," *Videre: A Journal of Computer Vision Research*, vol. 1, no. 3, 1999.

[18] J.-L. Lauriere, "A language and a program for stating and solving combinatorial problems," *Artificial Intelligence*, vol. 10, no. 1, pp. 29–127, 1978.

[19] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, no. 1, pp. 99–118, 1977.

[20] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[21] P. Wolfe, "The reduced gradient method," Jun. 1962, unpublished.

[22] ——, "Methods of nonlinear programming," in *Nonlinear Programming*, J. Abadie, Ed. Interscience, John Wiley, 1967, ch. 6, pp. 97–131.

[23] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2008, pp. 1–8.

[24] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[25] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–19, 2002.

[26] D. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, P. Winston, Ed. McGraw-Hill, 1975, pp. 19–91.

Nonstandard Interpretations of Probabilistic Programs for Efficient Inference

Anonymous Author(s)
Affiliation
Address
email

Abstract

Probabilistic programming languages allow modelers to specify a stochastic process using syntax that resembles modern programming languages. Because the program is in machine-readable format, a variety of techniques from compiler design and program analysis can be used to examine the structure of the distribution represented by the probabilistic program. We show how *nonstandard interpretations* of probabilistic programs can be used to craft efficient inference algorithms: information about the structure of a distribution (such as gradients or bounds) is generated as a monad-like side computation while executing the program. These interpretations can be easily coded using special-purpose objects and operator overloading. We implement two examples of nonstandard interpretations in two different languages, and use them as building blocks to construct inference algorithms: automatic differentiation, which enables gradient based methods, and provenance tracking, which enables efficient construction of global proposals.

1 Introduction

Probabilistic programming simplifies the development of probabilistic models by allowing modelers to specify a stochastic process using syntax that resembles modern programming languages. These languages permit arbitrary mixing of deterministic and stochastic elements, resulting in tremendous modeling flexibility. The resulting programs define probabilistic models that serve as prior distributions: running the (unconditional) program forward many times results in a distribution over execution traces, with each trace being a sample from the prior. Examples include BLOG [11], Bayesian Logic Programs [9] IBAL [14], Church [5], Stochastic Matlab [24], and HANSEI [10].

The primary challenge in developing such languages is scalable inference. Inference can be viewed as reasoning about the posterior distribution over execution traces conditioned on a particular program output, and is difficult because of the flexibility these languages present: in principle, an inference algorithm must behave reasonably for any program a user wishes to write. Sample-based MCMC algorithms are the state-of-the-art method, due to their simplicity, universality, and compositionality. But in probabilistic modeling more generally, efficient inference algorithms are designed by taking advantage of structure in distributions. How can we find structure in a distribution defined by a probabilistic program? A key observation is that some languages, such as Church and Stochastic Matlab, are defined in terms of an existing (non-probabilistic) language. Programs written in these languages may literally be executed in their native environments—suggesting that tools from program analysis and programming language theory can be leveraged to find and exploit structure in the program for inference, much as a compiler might find and exploit structure for performance.

Here, we show how *nonstandard interpretations* of probabilistic programs can help craft efficient inference algorithms. Information about the structure of a distribution (such as gradients, dependencies or bounds) is generated as a monad-like side computation while executing the program.

This extra information can be used to, for example, construct good MH proposals, or search efficiently for a local maximum. We focus on two such interpretations: automatic differentiation and provenance tracking, and show how they can be used as building blocks to construct efficient inference algorithms. We implement nonstandard interpretations in two different languages (Church and Stochastic Matlab), and experimentally demonstrate that while they typically incur some additional execution overhead, they dramatically improve inference performance.

2 Background and Related Work

Alg. 1: A Gaussian-Gamma mixture

```
1: for i=1:1000
2:     if ( rand > 0.5 )
3:         X(i) = randn;
4:     else
5:         X(i) = gammarnd;
6:     end;
7: end;
```

We begin by outlining our setup, following [24]. We define an unconditioned probabilistic program to be a parameterless function f with an arbitrary mix of stochastic and deterministic elements (hereafter, we will use the term function and program interchangeably). The function f may be written in any language, but our running example will be Matlab. We allow the function to be arbitrarily complex inside, using any additional functions, recursion, language constructs or external libraries it wishes. The only constraint is that the function must be self-contained, with no external side-effects which would impact the execution of the function from one run to another.

The stochastic elements of f must come from a set of known, fixed *elementary random primitives*, or ERPs. Complex distributions are constructed compositionally, using ERPs as building blocks. In Matlab, ERPs may be functions such as `rand` (sample uniformly from [0,1]) or `randn` (sample from a standard normal). Higher-order random primitives, such as nonparametric distributions, may also be defined, but must be fixed ahead of time. Formally, let $\mathcal{T}$ be the set of ERP types. We assume that each type $t \in \mathcal{T}$ is a parametric family of distributions $p_t(x|\theta_t)$, with parameters θ_t.

Now, consider what happens while executing f. As f is executed, it encounters a series of ERPs. Alg. 1 shows an example of a simple f written in Matlab with three syntactic ERPs: `rand`, `randn`, and `gammarnd`. During execution, depending on the return value of each call to `rand`, different paths will be taken through the program, and different ERPs will be encountered. We call this path an *execution trace*. A total of 2000 random choices will be made when executing this f.

Let $f_{k|x_1,\cdots,x_{k-1}}$ be the k'th ERP encountered while executing f, and let x_k be the value it returns. Note that the parameters passed to the k'th ERP may change depending on previous x_k's (indeed, its type may also change, as well as the total number of ERPs). We denote by x all of the random choices which are made by f, so f defines the probability distribution $p(x)$. In our example, $x \in \mathbb{R}^{2000}$. The probability $p(x)$ is the product of the probability of each individual ERP choice:

$$p(x) = \prod_{k=1}^{K} p_{t_k}(x_k|\theta_{t_k}, x_1, \cdots, x_{k-1}) \tag{1}$$

again noting explicitly that types and parameters may depend arbitrarily on previous random choices. To simplify notation, we will omit the conditioning on the values of previous ERPs, but again wish to emphasize that these dependencies are critical and cannot be ignored. By f_k, it should therefore be understood that we mean $f_{k|x_1,\cdots,x_{k-1}}$, and by $p_{t_k}(x_k|\theta_{t_k})$ we mean $p_{t_k}(x_k|\theta_{t_k}, x_1, \cdots, x_{k-1})$.

Generative functions as described above are, of course, easy to write. A much harder problem, and our goal in this paper, is to reason about the posterior conditional distribution $p(x|y)$, where we define y to be a subset of random choices which we condition on and (in an abuse of notation) x to be the remaining random choices. For example, we may condition f on the `X(i)`'s, and reason about the sequence of `rand`'s most likely to generate the `X(i)`'s. For the rest of this paper, we will drop y and simply refer to $p(x)$, but it should be understood that the goal is always to perform inference in conditional distributions.

2.1 Nonstandard Interpretations of Probabilistic Programs

With an outline of probabilistic programming in hand, we now turn to nonstandard interpretations. The idea of nonstandard interpretations originated in model theory and mathematical logic, where it was proposed that a set of axioms could be interpreted by different models. For example, differential geometry can be considered a nonstandard interpretation of classical arithmetic.

In programming, a nonstandard interpretation replaces the domain of the variables in the program with a new domain, and redefines the semantics of the operators in the program to be consistent with the new domain. This allows reuse of program syntax while implementing new functionality. For example, the expression "$a * b$" can be interpreted equally well if a and b are either scalars or matrices, but the "$*$" operator takes on different meanings. Practically, many useful nonstandard interpretations can be implemented with operator overloading: variables are redefined to be objects with operators that implement special functionality, such as tracing, reference counting, or profiling.

For the purposes of inference in probabilistic programs, we will augment each random choice x_k with additional side information s_k, and replace each x_k with the tuple $\langle x_k, s_k \rangle$. The native interpreter for the probabilistic program can then interpret the source code as a sequence of operations on these augmented data types. For a recent example of this, we refer the reader to [19].

3 Automatic Differentiation

For probabilistic models with many continuous-valued random variables, the gradient of the likelihood $\nabla_x p(x)$ provides local information that can significantly improve the properties of Monte Carlo inference algorithms. For instance Langevin Monte-Carlo [16] and Hamiltonian MCMC [13] use this gradient as part of a variable-augmentation technique (described below). We would like to be able to use gradients in the probabilistic program setting, but $p(x)$ is represented implicitly by the program. How can we compute its gradient? We use *automatic differentiation* (AD) [3, 6], a nonstandard interpretation that automatically constructs $\nabla_x p(x)$. The automatic nature of AD is critical because relieves the programmer from hand-computing derivatives for each model; moreover, some probabilistic programs dynamically create or delete random variables making simple closed-form expressions for the gradient very difficult to find.

AD is a technique for computing the gradient of a function f that, unlike finite differencing, computes an exact derivative at a point (up to machine precision). To do this AD relies on the chain rule to decompose the derivative of f into derivatives of its sub-functions: ultimately, known derivatives of elementary functions are composed together to yield the derivative of the complex function. This composition can be computed as a non-standard interpretation of the underlying real operations. In *forward mode* [23] AD this interpretation can be seen as extending each real value to the first two terms of its taylor expansion, overloading each real operator to operate on these real "polynomials". Because the derivatives of f at c can be extracted from the coefficients of ϵ in $f(c + \epsilon)$ [21], this allows computation of the gradient. *Reverse mode* [20] AD (which we use in our implementation) constructs an alternative non-standard interpretation by extending real values into "tapes" that capture the trace of the real computation which led to the primary value; this can be used to much more efficiently compute the gradient.

There are implementations of AD for many languages, including Scheme (e.g., [19]), FORTRAN (e.g., ADIFOR[2]), C (e.g., ADOL–C [7]), C++ (e.g., FADBAD++[1]), MATLAB (e.g., INTLAB [17]), and MAPLE (e.g., GRADIENT [12]). See `www.autodiff.org`.

3.1 Hamiltonian MCMC

To illustrate the power of AD in probabilistic programming, we build on Hamiltonian MCMC (HMC), and efficient algorithm whose popularity has been somewhat limited by the necessity of computing gradients—a difficult task for complex models. Neal [13] introduces HMC as an inference method which "produces distant proposals for the Metropolis algorithm, thereby avoiding the slow exploration of the state space that results from the diffusive behavior of simple random-walk proposals." HMC begins by augmenting the states space with "momentum

Alg. 2: Hamiltonian MCMC

1: repeat forever
2: **Gibbs step:**
3: Draw momentum $m \sim \mathcal{N}(0, \sigma^2)$
4: **Metropolis step:**
5: Start with current state (x, m)
6: Simulate Hamiltonian dynamics to give (x', m')
7: Accept w/ $p = \min[1, e^{(-H(x',m')+H(x,m))}]$
8: end;

```
(define (perlin-pt x y keypt power)
  (* 255 (sum (map (lambda (p2 pow)
                     (* pow (2d-interp (keypt (ceil (* p2 x)))
                                       (keypt (floor (* p2 x)))
                                       (keypt (ceil (* p2 y)))
                                       (keypt (floor (* p2 y))))))
                   powers-of-2 power))))
(define (perlin xs ys power)
  (let* ((keypt (mem (lambda (x y) (/ 1 (+ 1 (exp (- (gaussian 0.0 2.0)))))))))
    (map (lambda (x) (map (lambda (y) (perlin-pt x y keypt power)) xs)) ys)))
```

Figure 1: Code for the structured Perlin noise generator. `2d-interp` is B-spline interpolation.

variables" m. The distribution over this augmented space is $e^{H(x,m)}$, where the Hamiltonian function H decomposed into the sum of a potential energy term $U(x) = -\ln p(x)$ and a kinetic energy $K(m)$ which is usually taken to be Gaussian. Inference proceeds by alternating between a Gibbs step and Metropolis step: fixing the current state x, a new momentum m is sampled from the prior over m; then x and m are updated together by following a trajectory according to Hamiltonian dynamics. Discrete integration of Hamiltonian dynamics requires the gradient of H, and must be done with a symplectic (i.e. volume preserving) integrator (following [13] we use the Leapfrog method). While this is a complex computation, incorporating gradient information dramatically improves performance over vanilla random-walk style MH moves (such as Gaussian drift kernels), and its statistical efficiency also scales much better with dimensionality than simpler methods [13].

It would also be straightforward to use AD to compute higher derivates (though this would introduce super-linear overhead). For instance, Hessian matrices could be used to construct blocked Metropolis moves [8], to construct proposals based on Newton's method [15], or as part of Riemannian manifold methods [4].

3.2 Experiments and Results

We implemented HMC by extending Bher [24], a lightweight implementation of the Church language which provides simple, but universal, MH inference. We used used an implementation of AD based on [19], that uses hygienic operator overloading to do both forward and reverse mode AD for Scheme (the target language of the Bher compiler).

The goal is to compute $\nabla_x p(x)$. By Eq. 1, $p(x)$ is the product of the individual choices made by each x_i (though each probability can depend on previous choices, through the program evaluation). To compute $p(x)$, Bher executes the corresponding program, accumulating likelihoods. Each time a continuous ERP is created or retrieved, we wrap it in a "tape" object which is used to track gradient information; as the likelihood $p(x)$ is computed, these tapes flow through the program and through appropriately overloaded operators, resulting in a dependency graph for the real portion of the computation. The gradient is then computed in reverse mode, by "back-propagating" along this graph. We implement an HMC kernel by using this gradient in the leapfrog integrator. Since program states may contain a combination of discrete and continuous ERPs, we use an overall cycle kernel which alternates between standard MH kernel for individual discrete random variables and the HMC kernel for all continuous random choices. To decrease burn-in time, we initialize the sampler by using annealed gradient ascent (again implemented using AD).

We ran two sets of experiments which illustrate two different benefits of HMC with AD: automated gradients of complex code, and good statistical efficiency.

Structured Perlin noise generation. Our first experiment uses HMC to generate modified Perlin noise with soft symmetry structure. Perlin noise is a procedural texture used by computer graphics artists to add realism to natural textures such as clouds, grass or tree bark. We generate Perlin-like noise by layering octaves of random but smoothly varying functions. We condition the result on approximate diagonal symmetry, forcing the resulting image to incorporate additional structure without otherwise skewing the statistics of the image. Note that the MAP solution for this problem is uninteresting, as it is a uniform image; it is the variations around the MAP that provide rich texture. We generated 48x48 images; the model had roughly 1000 variables.

Fig. 2 shows the result via typical samples generated by HMC, where the approximate symmetry is clearly visible. A code snippet demonstrating the complexity of the calculations is shown in Fig. 1;

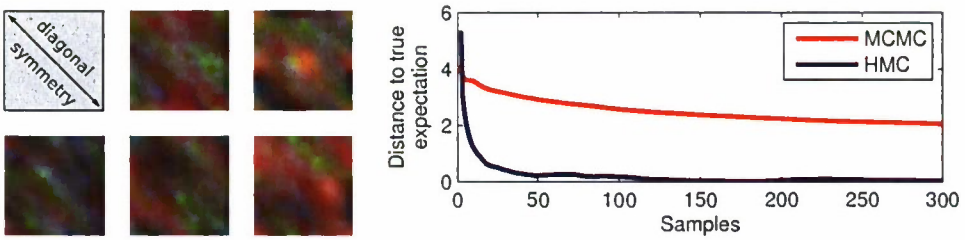


Figure 2: On the left: samples from the structured Perlin noise generator. On the right: convergence of expected mean for a draw from a 3D spherical Gaussian conditioned on lying on a line.

this experiment illustrates how the automatic nature of the gradients is most helpful, as it would be time consuming to compute these gradients by hand—particularly since we are free to condition using any function of the image.

Complex conditioning. For our second example, we demonstrate the improved statistical efficiency of the samples generated by HMC versus Bher's standard MCMC algorithm. In this task, the goal is to sample points from a complex 3-dimensional distribution. We construct this by starting with a simple Gaussian mixture model prior, but then sample points that are noisily conditioned to be on a line running through $\mathbb{R}^3$. This conditioner creates complex interactions with the prior to yield a smooth, but strongly coupled, energy landscape.

Fig. 2 shows results comparing our HMC implementation with Bher's standard MCMC inference engine. The x-axis denotes samples, while the y-axis denotes the convergence of an estimator of certain marginal statistics of the samples. We see that this estimator converges much faster for HMC, implying that the samples which are generated are less autocorrelated – affirming that HMC is indeed making better distal moves. HMC is about 5x slower than MCMC for this experiment, but the overhead is justified by the significant improvement in the statistical quality of the samples.

4 Provenance Tracking for Fine-Grained Dynamic Dependency Analysis

One reason gradient based inference algorithms are effective is that the chain rule of derivatives provides a principled way to compositionally backpropagate information from the data up to the proposal variables. But gradients, and the chain rule, are only defined for continuous variables. Is there a corresponding structure for discrete choices? We now introduce a new nonstandard interpretation based on provenance tracking (PT). In programming language theory, the provenance of a variable is the history of variables and computations that combined to form its value. In probabilistic programming, we propose to use this to track fine-grained dependency information between random values and intermediate computations as they combine to form a likelihood.

Importantly, the provenance information is collected for a particular value x of the probabilistic program, which can be useful for models with sparse, dynamic dependencies among variables. This can provide more detailed information than, say, a graphical model: in a graphical model, conditional independencies must hold for every value of variables in the model, but in practice, for a specific value, the dependencies may be sparser than the graph indicates. An example of this is shown in Alg. 4, where a simple renderer renders a triangle mesh into an image. Vertices in the mesh can move arbitrarily, so there is *some* value for each vertex such that every triangle could be rendered to any pixel, but for any *particular* set of vertex values, each triangle affects a small number of pixels.

4.1 Defining and Implementing Provenance Tracking

Like AD, PT can be implemented with operator overloading. Because provenance information is much coarser than gradient information, the operators in PT objects have a particularly simple form; most program expressions can be covered by considering a few cases. Let X denote the set $\{x_i\}$ of all (not necessarily random) variables in a program. Let $R(x) \subset X$ define the provenance of a variable x. Given $R(x)$, the provenance of expressions involving x can be computed by breaking

down expressions into a sequence of unary operations, binary operations, and function applications. Constants have empty provenances.

Let x and y be expressions in the program (consisting of an arbitrary mix of variables, constants, functions and operators). For a binary operation $x \odot y$, the provenance $R(x \odot y)$ of the result is defined to be $R(x \odot y) = R(x) \cup R(y)$. Similarly, for a unary operation, the provenance $R(\odot x) = R(x)$. For assignments, $x = y \Rightarrow R(x) = R(y)$. For a function, $R(f(x, y, ...))$ may be computed by examining the expressions within f; a worst-case approximation is $R(f(x, y, ...)) = R(x) \cup R(y) \cdots$. A few special cases are also worth noting. Strictly speaking, the previous rules track a superset of provenance information because some functions and operations are constant for certain inputs. In the case of multiplication, $x * 0 = 0$, so $R(x * 0) = \{\}$. Accounting for this gives tighter provenances, implying, for example, that special considerations apply to sparse linear algebra.

In the case of probabilistic programming, recall that random variables (or ERPs) are represented as stochastic functions f_i that accept parameters θ_i. Whenever a random variable is conditioned, the output of the corresponding f_i is fixed; thus, while the *likelihood* of a particular output of f_i depends on θ_i, the *specific output* of f_i does not. For the purposes inference, therefore, $R(f_i(\theta_i)) = \{\}$.

4.2 Using Provenance Tracking as Part of Inference

Provenance information could be used in many ways. Here, we illustrate one use: to help construct good block proposals for MH inference. Our basic idea is to construct a good global proposal by starting with a random global proposal (which is unlikely to be good) and then inhibiting the bad parts. We now go through the steps of our algorithm, which is summarized in Fig. 3. Let x denote a state of the probabilistic program (equivalently, we will also use x to refer to the set of random variables instantiated at this state). For notational simplicity we will assume we wish to construct a proposal for all variables in x (but extending to the case of proposing to a subset is straightforward).

In step (1), we compute $p(x)$ using PT to track how each x_i influences the overall likelihood $p(x)$. Let $D(x_i; x) \subset x$ denote the "descendants" of variable x_i, meaning all ERPs whos likelihood x_i impacted. In step (2), we propose $x' \sim q(x'|x)$, and in step (3) we use PT to compute $p(x')$, again tracking dependents $D(x_i; x') \subset x'$. In step (4) we let $D(i) = D(x_i; x) \cup D(x_i; x')$ be the joint set of ERPs that x_i influences in either state x or x'. In step (5-6) we use $D(i)$, $p(x)$ and $p(x')$ to estimate the amount by which each constituent element x'_i in the proposal changed the likelihood. We assign "credit" to each x'_i *as if it were the only proposal* – that is, we assume that if, for example, the likelihood went up, it was entirely due to the change in x_i. Of course, the variables' effects are not truly independent; this is a fully-factored approximation to those effects. We define the approximate credit as $c(i) = \frac{p(x'_i)p(x'_{D(i)})}{p(x_i)p(x_{D(i)})}$, where we define $p(x_{D(i)})$ to be the likelihood of *only* the subset of variables that x_i impacted. Based on the credit assigned to each x'_i, we construct a new proposal x^M by composing x'_i's with high credit. We start by setting $x^M = x$. We then compute a standard MH ratio for each x_i, setting $x_i^M = x'_i$ with probability $\alpha(x_i^M|x, x') = \min\{1, c(i)\}$. The overall proposal probability for this "mixing" step is therefore $\alpha(x^M|x, x') = \prod_i \alpha(x_i^M|x, x')$. In steps (7-8) we compute the overall forward and reverse transition probabilities as explained below. Finally, in step (9) we accept or reject the overall proposal.

Thus, we allow the likelihood to "vote" in a fine-grained way for which proposals seemed to be good and which seemed to be bad. Note that the method coarsely interpolates between two end-cases: if the variables x really are fully factored, the method reduces to making independent proposals which are accepted or rejected independently. If the dependencies are dense, it reduces to making a random global proposal. The method therefore critically relies on sparsity to be effective.

There are some subtleties to the implementation. In general, $D(x_i; x) \neq D(x_i; x')$; the proposal may have added dependents or removed dependents. To ensure that all terms in the likelihood are accounted for, we assign credit to x_i based on the change in likelihood for all $x \in D(i)$ (but better proposals could be made by accounting for higher-order interactions among variables). Also note that the proposal probability $q(x'|x)$ must factorize as $\prod_i q(x'_i|x)$, because it must be able to score $q(x^{-M}|x^M)$ (see below), but x^M and x^{-M} are constructed compositionally from bits of x and x'.

We analyze the provenance tracking algorithm as a proposal in an MH algorithm, in a manner analogous to delayed rejection [22]: we show how the kernel moves through a reversible sequence

Alg. 3: The provenance tracking algorithm

1: Compute $p(x)$, tracking $D(x_i; x)$
2: Propose $x' \sim q(x'|x)$
3: Compute $p(x')$, tracking $D(x_i; x')$
4: Let $D(i) = D(x_i; x) \cup D(x_i; x')$
5: Generate x^M by accepting each x'_i independently:
6: Accept x'_i with prob. $\min\left\{1, \frac{p(x'_i)p(x'_{D(i)})}{p(x_i)p(x_{D(i)})}\right\}$
7: Compute $f = q(x'|x)\alpha(x^M|x, x')$
8: Compute $r = q(x^{-M}|x^M)\alpha(x'|x^M, x^{-M})$
9: Accept x^M with prob. $\min\left\{1, \frac{p(x^M)f}{p(x)r}\right\}$

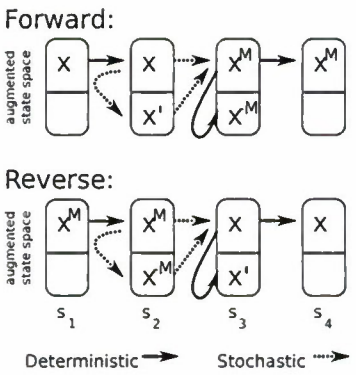


Figure 3: The provenance tracking algorithm

of states, and compute the transition probabilities of this sequence. To do this, we consider an augmented state space. The kernel goes through four stages, shown in Fig. 3. It begins in $s_1 = (x)$, from which it proposes x' from $q(x'|x)$, to form the joint space $s_2 = (x, x')$. Next, it "mixes" x and x' together via individual accept/reject decisions to create a new state x^M, and we define x^{-M} as the state obtained if all accept/reject decisions were reversed. This transition probability is $\alpha(x^M|x, x')$ (note that given x^M, x^{-M} is computed deterministically). This leads to the third state in the kernel, $s_3 = (x^M, x^{-M})$. From here, we deterministically transition to $s_4 = (x^M)$. The forward probability is thus given by $p(s_2|s_1)p(s_3|s_2)p(s_4|s_3) = q(x'|x)\alpha(x^M|x, x')$. The reverse probability is $p(s_3|s_4)p(s_2|s_3)p(s_1|s_2) = q(x^{-M}|x^M)\alpha(x'|x^M, x^{-M})$.

4.3 Experiments and Results

We implemented provenance tracking and in Stochastic Matlab [24] by leveraging Matlab's object oriented capabilities, which provides full operator overloading. We tested on four tasks: a Bayesian "mesh induction" task, a small QMR problem, probabilistic matrix factorization [18] and an integer-valued variant of PMF. We measured performance by examining likelihood as a function of wallclock time; an important property of the provenance tracking algorithm is that it can help mitigate constant factors affecting inference performance.

Bayesian mesh induction. The BMI task is simple: given a prior distributions over meshes and a target image, sample a mesh which, when rendered, looks like the target image. The prior is a Gaussian centered around a "mean mesh," which is a perfect sphere; Gaussian noise is added to each vertex to deform the mesh. The mesh has 2,500 vertices in $\mathbb{R}^3$, for a total of 7,500 parameters. The model is shown in Alg. 4. The rendering function is a custom OpenGL renderer implemented as a MEX function. No gradients are available for this renderer, but it is reasonably easy to augment it with provenance information: we use shaders to record the vertices of the triangle that were responsible for each pixel. This allows us to make proposals to mesh vertices, while assigning credit based on pixel likelihoods.

Alg. 4: Bayesian Mesh Induction

```
1: function X = bmi( base_mesh )
2:     mesh = base_mesh + randn;
3:     img = render( mesh );
4:     X = img + randn;
5: end;
```

Results for this task are shown in Fig. 4 ("Face"). Note that even though the renderer is quite fast, MCMC with simple proposals fails dramatically: after proposing a change to a single variable, it must re-render the image in order to compute the likelihood. In contrast, making large, global proposals is very effective. Fig. 4 (top) shows a sequence of images representing burn-in of the model as it starts from the initial condition and samples its way towards regions of high likelihood. *An anonymous video demonstrating the results is available at http://www.liftothers.org/face.avi.*

QMR. The QMR model is a bipartite, binary model relating diseases (hidden) to symptoms (observed) using a log-linear noisy-or model. Base rates on diseases can be quite low, so "explaining away" can cause poor mixing. Here, MCMC with provenance tracking is effective: it finds high-likelihood solutions quickly, again outperforming naive MCMC.

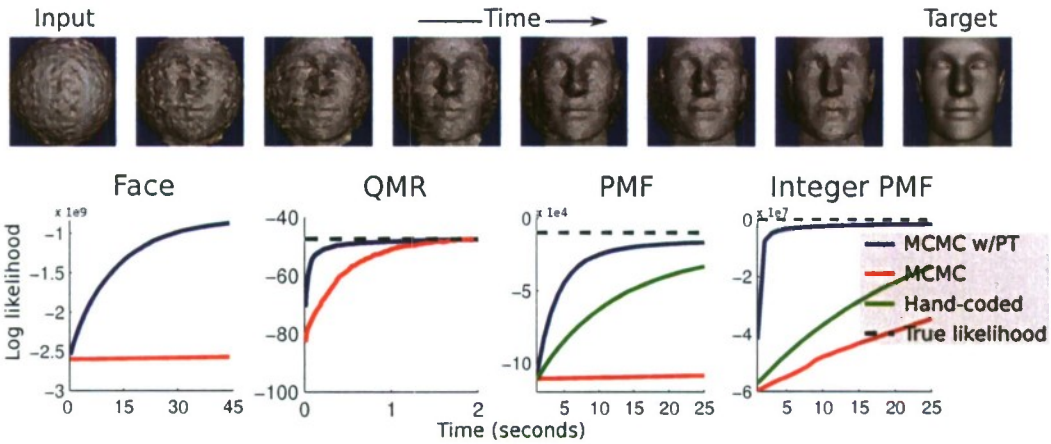


Figure 4: Top: Frames from the face task. Bottom: results on Face, QMR, PMF and Integer PMF.

Probabilistic Matrix Factorization. For the PMF task, we factored a matrix $A \in \mathbb{R}^{1000x1000}$ with 99% sparsity. PMF places a Gaussian prior over two matrices, $U \in \mathbb{R}^{1000x10}$ and $V \in \mathbb{R}^{1000x10}$, for a total of 20,000 parameters. The model assumes that $A_{ij} \sim \mathcal{N}(U_i V_j^T, 1)$. In Fig. 4, we see that MCMC with provenance tracking is able to find regions of much higher likelihood much more quickly than naive MCMC. We also compared to an efficient hand-coded MCMC sampler which is capable of making, scoring and accepting/rejecting about 20,000 proposals per second. Interestingly, MCMC with provenance tracking is more efficient than the hand-coded sampler, presumably because of the economies of scale that come with making global proposals.

Integer Probabilistic Matrix Factorization. The Integer PMF task is like ordinary PMF, except that every entry in U and V is constrained to be an integer between 1 and 10. These constraints imply that no gradients exist. Empirically, this does not seem to matter for the efficiency of the algorithm relative to standard MCMC: in Fig. 4 we again see dramatic performance improvements over the baseline Stochastic Matlab sampler and the hand-coded sampler.

5 Conclusions

We have shown how nonstandard interpretations of probabilistic programs can be used to extract structural information about a distribution, and how this information can be used as part of a variety of inference algorithms. The information can take the form of gradients, Hessians, fine-grained dependencies, or bounds. Empirically, we have implemented two such interpretations and demonstrated how this information can be used to find regions of high likelihood quickly, and how it can be used to generate samples with improved statistical properties versus random-walk style MCMC. There are doubtless other types of interpretations which could provide additional information. For example, interval arithmetic [17] (or its higher-order generalizations, such as affine arithmetic) could be used to provide bounds, or, in conjunction with recursive bisection, could be used as part of MAP inference, line searches, or adaptive importance sampling.

Each of these interpretations can be used alone or in concert with each other; one of the advantages of the probabilistic programming framework is the clean separation of models and inference algorithms, making it easy to explore combinations of inference algorithms for complex models. For example, perhaps provenance tracking-based proposals for discrete variables can be interlaced with HMC proposals for continuous variables. Or, Neal [13] points out that for HMC the true likelihood does not need to be used while integrating along a trajectory; a simpler likelihood may be used instead, as long as the true likelihood is used for the final accept/reject calculation. This suggests learning an approximate likelihood that can be cheaply computed; perhaps the interval-based approximations can be used to construct this on-demand.

More generally, this work begins to illuminate the close connections between probabilistic inference and programming language theory. It is likely that other techniques from compiler design and program analysis could be fruitfully applied to inference problems in probabilistic programs.

References

[1] C. Bendtsen and O. Stauning. FADBAD, a flexible C++ package for automatic differentiation. Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, Aug. 1996.

[2] C. H. Bischof, A. Carle, G. F. Corliss, A. Griewank, and P. D. Hovland. ADIFOR: Generating derivative codes from Fortran programs. *Scientific Programming*, 1(1):11–29, 1992.

[3] G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann. *Automatic Differentiation: From Simulation to Optimization*. Springer-Verlag, New York, NY, 2001.

[4] M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. R. Statist. Soc. B*, 73(2):123214, 2011.

[5] N. Goodman, V. Mansinghka, D. Roy, K. Bonawitz, and J. Tenenbaum. Church: a language for generative models. In *Uncertainty in Artificial Intelligence (UAI)*, 2008.

[6] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Applied Mathematics. SIAM, 2000.

[7] A. Griewank, D. Juedes, and J. Utke. ADOL-C, a package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Software*, 22(2):131–67, 1996.

[8] E. Herbst. Gradient and Hessian-based MCMC for DSGE models (job market paper), 2010.

[9] K. Kersting and L. D. Raedt. Bayesian logic programming: Theory and tool. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[10] O. Kiselyov and C. Shan. Embedded probabilistic programming. In *Domain-Specific Languages*, pages 360–384, 2009.

[11] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. BLOG: Probabilistic models with unknown objects. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1352–1359, 2005.

[12] M. B. Monagan and W. M. Neuenschwander. GRADIENT: Algorithmic differentiation in Maple. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 68–76, July 1993.

[13] R. M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte-Carlo (Steve Brooks, Andrew Gelman, Galin Jones and Xiao-Li Meng, Eds.)*, 2010.

[14] A. Pfeffer. IBAL: A probabilistic rational programming language. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 733–740. Morgan Kaufmann Publ., 2001.

[15] Y. Qi and T. P. Minka. Hessian-based Markov chain Monte-Carlo algorithms (unpublished manuscript), 2002.

[16] P. J. Rossky, J. D. Doll, and H. L. Friedman. Brownian dynamics as smart monte carlo simulation. *Journal of Chemical Physics*, 69:4628–4633, 1978.

[17] S. Rump. INTLAB - INTerval LABoratory. In *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999.

[18] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Neural Information Processing Systems (NIPS)*, 2008.

[19] J. M. Siskind and B. A. Pearlmutter. First-class nonstandard interpretations by opening closures. In *POPL*, 2007.

[20] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Jan. 1980.

[21] B. Taylor. *Methodus Incrementorum Directa et Inversa*. London, 1715.

[22] L. Tierney and A. Mira. Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine*, 18:2507–2515, 1999.

[23] R. E. Wengert. A simple automatic derivative evaluation program. *Commun. ACM*, 7(8):463–4, 1964.

[24] D. Wingate, A. Stuhlmueller, and N. D. Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Seeing Unseeability to See the Unseeable

Siddharth Narayanaswamy, Andrei Barbu, and Jeffrey Mark Siskind

Abstract— **We present a framework that allows an observer to determine occluded portions of a structure by finding the maximum-likelihood estimate of those occluded portions consistent with visible image evidence and a consistency model. Doing this requires determining which portions of the structure are occluded in the first place. Since each process relies on the other, we determine a solution to both problems in tandem. We extend our framework to determine confidence of one's assessment of which portions of an observed structure are occluded, and the estimate of that occluded structure, by determining the sensitivity of one's assessment to potential new observations. We further extend our framework to determine a robotic action whose execution would allow a new observation that would maximally increase one's confidence.**

I. Introduction

> [T]here are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns–the ones we don't know we don't know.
>
> Donald Rumsfeld (12 February 2002)

People exhibit the uncanny ability to see the unseeable. The colloquial exhortation *You have eyes in the back of your head!* expresses the assessment that someone is making correct judgements as if they could see what is behind them, but obviously cannot. People regularly determine the properties of occluded portions of objects from observations of visible portions of those objects using general world knowledge about the consistency of object properties. Psychologists have demonstrated that the world knowledge that can influence perception can be high level, abstract, and symbolic, and not just related to low-level image properties such as object class, shape, color, motion, and texture. For example, Freyd et al. [7] showed that physical forces, such as gravity, and whether such forces are in equilibrium, due to support and attachment relations, influences visual perception of object location in adults. Baillargeon [1], [2] showed that knowledge of substantiality, the fact that solid objects cannot interpenetrate, influences visual object perception in young infants. Streri et al. [20] showed that knowledge about object rigidity influences both visual and haptic perception of those objects in young infants. Moreover, such influence is cross modal: observable haptic perception influences visual perception of unobservable properties and observable visual perception influences haptic perception of unobservable properties. Wynn [21] showed that material properties of objects, such as whether they are countable or mass substances, along with abstract properties, such as the number of countable objects and the quantity of mass substances, and how they are transferred between containers, influences visual perception in young infants. Similar results exist for many physical properties such as relative mass, momentum, etc. These results demonstrate that people can easily integrate information from multiple sources together with world knowledge to see the unseeable.

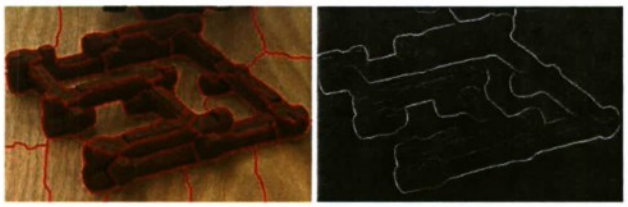
People so regularly invoke the ability to see the unseeable that we often don't realize that we do so. If you observe a person entering the front door of a house and later see them appear from behind the house without seeing them exit a door, you easily see the unseeable and conclude that there must be an unseen door to the house. But if one later opens the garage door or the curtain covering a large living-room bay window in the front of the house so that you see through the house and see the back door you no longer need to invoke the ability to see the unseeable. A more subtle question then arises: when must you invoke the ability to see the unseeable? In other words how can you see unseeability, the inability to see? This question becomes particularly thorny since, as we will see, it can involve a chicken-and-egg problem: seeing the unseen can require seeing the unseeability of the unseen and seeing the unseeability of the unseen can require seeing the unseen.

The ability to see unseeability and to see the unseeable can further dramatically influence human behavior. We regularly and unconsciously move our heads and use our hands to open containers to render seeable what was previously unseeable. To realize that we need to do so in the first place, we must first see the unseeability of what we can't see. Then we must determine how to best use our collective perceptual, motor, and reasoning affordances to remedy the perceptual deficiency.

We present a general computational framework for seeing unseeability to see the unseeable. We formulate and evaluate a particular instantiation of this general framework in the context of a restricted domain, namely LINCOLN LOGS, a children's assembly toy where one constructs assemblies from a small inventory of component logs. The two relevant aspects of this domain that facilitate its use for investigating our general computational framework are (a) that LINCOLN LOG assemblies suffer from massive occlusion and (b) that a simple but rich expression of world knowledge, in the form of constraints on valid assemblies, can mitigate the effects of such occlusion. While LINCOLN LOGS are a children's toy,

http://engineering.purdue.edu/~qobi/icra2012

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907, USA {snarayan,abarbu,qobi}@purdue.edu



(a) (b)

Fig. 1. (a) A state-of-the-art segmentation method, Normalized Cut [18], does not segment out the log parts. (b) A state-of-the-art edge detector, GPB [14], does not reliably find edges separating adjacent logs or log ends.

this domain is far from a toy when it comes to computer vision. The task of structure estimation, determining, from an image, the correct combination of component logs used to construct an assembly and how they are combined, is well beyond state-of-the-art methods in computer vision. We have not found any general-purpose image segmentation methods that can determine the image boundaries of the visible component logs (see Fig. 1a). Moreover, the uniform matte color and texture of the logs, together with the fact that logs are placed in close proximity and the fact that the majority of any structure is in self shadow means every edge-detection method that we have tried fails to find the boundaries between adjacent logs (see Fig. 1b). This is even before one considers occlusion, which only makes matters worse.

Not only is the computer-vision problem for this domain immensely difficult, the computational problem is rich as well. We present methods for seeing the unseeable (in section II) and seeing unseeability (in section III) based on precise computation of the maximum-likelihood structure estimate conditioned on world knowledge that marginalizes over image evidence. We further present (in section IV) a rational basis for determining confidence in one's structure estimate despite unseeability based on precise computation of the amount of evidence needed to override a uniform prior on the unseeable. And we finally present (in section V) an active-vision decision-making process for determining rational behavior in the presence of unseeability based on precise computation of which of several available perception-enhancing actions one should take to maximally improve the confidence in one's structure estimate. We offer experimental evaluation of each of these methods in section VI, compare against related work in section VII, and conclude with a discussion of potential extensions in section VIII.

II. Structure Estimation

In previous work we [19] presented an approach for using a visual language model for improving recognition accuracy on compositional visual structures in a generative visual domain, over the raw recognition rate of the part detectors—by analogy to the way speech recognizers use a human language model to improve recognition accuracy on utterances in a generative linguistic domain, over the raw recognition rate of the phoneme detectors. In this approach, a complex object is constructed out of a collection of parts taken from a small part inventory. A language model, in the form of a stochastic constraint-satisfaction problem (CSP) [11], characterizes the constrained way object parts can combine to yield a whole object and significantly improves the recognition rate of the whole structure over the infinitesimally small recognition rate that would result from unconstrained application of the unreliable part detectors. Unlike the speech-recognition domain, where (except for coarticulation) there is acoustic evidence for all phonemes, in the visual domain there may be components with no image evidence due to occlusion. A novel aspect of applying a language model in the visual domain instead of the linguistic domain is that the language model can additionally help in recovering occluded information.

This approach was demonstrated in the domain of Lincoln Logs, a children's assembly toy with a small part inventory, namely, 1-notch, 2-notch, and 3-notch logs, whose CAD models are provided to the system. In this domain, a grammatical Lincoln Log structure contains logs that are parallel to the work surface and organized on alternating layers oriented in orthogonal directions. Logs on each layer are mutually parallel with even spacing, thereby imposing a symbolic grid on the Lincoln Log assembly. The symbolic grid positions $q = (i, j, k)$ refer to points along log medial axes at notch centers. One can determine the camera-relative pose of this symbolic grid without any knowledge of the assembly structure by fitting the pose to the two predominant directions of image edges that result from the projection of the logs to the image plane.

Each grid position may be either unoccupied, denoted by $\emptyset$, or occupied with the n^{th} notch, counting from zero, of a log with m notches, denoted by (m,n). Estimating the structure of an assembly reduces to determining the occupancy at each grid position, one of the seven possibilities: $\emptyset$, $(1,0)$, $(2,0)$, $(2,1)$, $(3,0)$, $(3,1)$, and $(3,2)$. This is done by constructing a discrete random variable Z_q for each grid position q that ranges over these seven possibilities, mutually constraining these random variables together with other random variables that characterize the image evidence for the component logs using the language model, and finding a maximum-likelihood consistent estimate to the random variables Z_q.

Several forms of image evidence are considered for the component logs. Lincoln Logs, being cylindrical parts, generate two predominant image (log) features: ellipses that result from the perspective projection of circular log ends and line segments that result from the perspective projection of cylindrical walls. The former are referred to as *log ends* and the latter as *log segments*. Log ends can potentially appear only at a fixed distance on either side of a grid position. Boolean random variables Z_q^+ and Z_q^- are constructed to encode the presence or absence of a log end at such positions. There are two kinds of log segments: those corresponding to the portion of a log between two notches and those corresponding to the portions of a log end that extend in front of or behind the two most extreme notches. Given this, three Boolean random variables Z_q^u, Z_q^v, and Z_q^w are constructed for each grid position q that encode the presence or absence of such log segments for the bottoms of logs, i.e. log segments

between a grid position and the adjacent grid position below.

A stochastic CSP encodes the validity of an assembly. Image evidence imposes priors on the random variables Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w and structure estimation is performed by finding a maximum-likelihood solution to this stochastic CSP. When formulating the constraints, the adjacent grid position below q is referred to as $b(q)$ and the adjacent grid position further from the origin along the direction of the grid lines for the layer of q is referred to as $n(q)$. Ignoring boundary conditions at the perimeter of the grid, the grammar of LINCOLN LOGS can be formulated as the following constraints:

a) 2-notch logs occupy two adjacent grid points
b) 3-notch logs occupy three adjacent grid points
c) 1- and 2-notch logs must be supported at all notches
d) 3-notch logs must be supported in at least 2 notches
e) log ends must be at the ends of logs
f) short log segments indicate occupancy above or below
g) long log segments indicate presence of a multi-notch log above or below

Boundary conditions are handled by stipulating that the grid positions beyond the perimeter are unoccupied, enforcing the support requirement (constraints c–d) only at layers above the lowest layer, and enforcing log-segment constraints (f–g) for the layer above the top of the structure. Structure estimation is performed by first establishing priors over the random variables Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w that correspond to log features using image evidence and establishing a uniform prior over the random variables Z_q that correspond to the latent structure. This induces a probability distribution over the joint support of these random variables. The random variables that correspond to log features are marginalized and the resulting marginal distribution is conditioned on the language model Φ. Finally, the assignment to the collection, $\mathbf{Z}$, of random variables Z_q, that maximizes this conditional marginal probability is computed.

$$\underset{\mathbf{Z}}{\operatorname{argmax}} \sum_{\substack{\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w \\ \Phi[\mathbf{Z},\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w]}} \Pr\left(\mathbf{Z},\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w\right)$$

While, in principle, this method can determine the conditional probability distribution over consistent structures given image evidence, doing so is combinatorially intractable. The conditional marginalization process is made tractable by pruning assignments to the random variables that violate the grammar Φ using arc consistency [13]. The maximization process is made tractable by using a branch-and-bound algorithm [10] that maintains upper and lower bounds on the maximal conditional marginal probability. Thus instead of determining the distribution over structures, this yields a single most-likely consistent structure given the image evidence, along with its probability.

III. VISIBILITY ESTIMATION

Image evidence for the presence or absence of each log feature is obtained independently. Each log feature corresponds to a unique local image property when projected to the image plane under the known camera-relative pose. A prior over the random variable associated with a specific log feature can be determined with a detector that is focused on the expected location and shape of that feature in the image given the projection. This assumes that the specific log feature is visible in the image, and not occluded by portions of the structure between the camera and that log feature. When the log feature f, a member of the set $\{+,-,u,v,w\}$ of the five feature classes defined above, at a position q, is not visible, the prior can be taken as uniform, allowing the constraints in the grammar to fill in unknown information. We represent the visibility of a feature by the boolean variable V_q^f.

$$\begin{array}{ll} \Pr(Z_q^f = \textbf{true}) \propto \text{image evidence} & \text{when } V_q^f = \textbf{true} \\ \Pr(Z_q^f = \textbf{false}) = \frac{1}{2} & \text{otherwise} \end{array}$$

In order to do so, it is necessary to know which log features are visible and which are occluded so that image evidence is only applied to construct a prior on visible log features and a uniform prior is constructed for occluded log features. Thus, in Rumsfeld's terminology, one needs to know the known unknowns in order to determine the unknowns. This creates a chicken-and-egg problem. To determine whether a particular log feature is visible, one must know the composition of the structure between that feature and the camera and, to determine the structure composition, one must know which log features are visible. While we earlier [19] demonstrated successful automatic determination of log occupancy at occluded log positions, we could only do so given manual annotation of log-feature visibility. In other words, while earlier we were able to automatically infer Z_q, it required manual annotation of V_q^f. Further, determining V_q^f required knowledge of Z_q.

We extend this prior work [19] to automatically determine visibility of log features in tandem with log occupancy. Our novel contribution in this section is mutual automatic determination of both Z_q and V_q^f. We solve the chicken-and-egg problem inherent in doing so with an iterative algorithm reminiscent of expectation maximization (EM) [4]–[6]. We start with an initial estimate of the visibility of each log feature. We then apply the structure estimation procedure developed in previous work [19] to estimate the occupancy of each symbolic grid position. We then use the estimated structure to recompute a new estimate of log-feature visibility, and iterate this process until a fixpoint is reached. There are two crucial components of this process: determining the initial log-feature visibility estimate and reestimating log-feature visibility from an estimate of structure.

We determine the initial log-feature visibility estimate (i.e. V_q^f) by assuming that the structure is a rectangular prism whose top face and two camera-facing front faces are completely visible. In this initial estimate, log features on these three faces are visible and log features elsewhere are not. We use the camera-relative pose of the symbolic grid (which can be determined without any knowledge of the structure) together with maximal extent of each of the three symbolic grid axes (i.e., three small integers which

are currently specified manually) to determine the visible faces. This is done as follows. We determine the image positions for four corners of the base of this rectangular prism: the origin $(0,0,0)$ of the symbolic grid, the two extreme points $(i_{\max},0,0)$ and $(0,0,k_{\max})$ of the two horizontal axes in the symbolic grid, and the symbolic grid point $(i_{\max},0,k_{\max})$. We select the bottommost three such image positions as they correspond to the endpoints of the lower edges of the two frontal faces. It is possible, however, that one of these faces is (nearly) parallel to the camera axis and thus invisible. We determine that this is the case when the angle subtended by the two lower edges previously determined is less than 110° and discard the face whose lower edge has minimal image width.

We update the log-feature visibility estimate from a structure estimate by rendering the structure in the context of the known camera-relative pose of the symbolic grid. When rendering the structure, we approximate each log as the bounding cylinder of its CAD model. We characterize each log feature with a fixed number of points, equally spaced around circular log ends or along linear log segments and trace a ray from each such point's 3D position to the camera center, asking whether that ray intersects some bounding cylinder for a log in the estimated structure. We take a log feature to be occluded when 60% or more of such rays intersect logs in the estimated structure. Our method is largely insensitive to the particular value of this threshold. It only must be sufficiently low to label log features as invisible when they actually are invisible. Structure estimation is not adversely affected by a moderate number of log features that are incorrectly labeled as invisible when they are actually visible because it can use the grammar to determine occupancy of grid positions that correspond to such log features.

We can perform such rendering efficiently by rasterization. For each log feature, we begin with an empty bitmap. We iterate over each log feature and each occupied grid position that lies between that log feature and the camera center and render a projection of the bounding cylinder of the log at that grid position on the bitmap. This renders all possible occluders for each log feature allowing one to determine visibility by counting the rendered pixels at points in the bitmap that correspond to the projected rays.

The above process might not reach a fixpoint and instead may enter a finite loop of pairs of visibility and structure estimates. In practice, this process either reaches a fixpoint within three to four iterations or enters a loop of length two within three to four iterations, making loop detection straightforward. When a loop is detected, we select the structure in the loop with the highest probability estimate.

IV. Structure-Estimation Confidence

While the structure estimation process [19] can determine the occupancy of a small number of grid positions when only a single set of occupancy values is consistent with the grammar and the image evidence, it is not clairvoyant; it cannot determine the structure of an assembly when a large part of that assembly is occluded and many different possible structures are consistent with the image evidence. In this case, we again have an issue of unknowns vs. known unknowns: how can one determine one's confidence in one's structure estimation. If we could determine the conditional distribution over consistent structures given image evidence, $P(\mathbf{Z}|I)$, we could take the entropy of this distribution, $H(\mathbf{Z}|I)$, as a measure of confidence. However, as discussed previously, it is intractable to compute this distribution and further intractable to compute its entropy. Thus we adopt an alternate means of measuring confidence in the result of the structure-estimation process.

Given a visibility estimate, V_q^f, a structure estimate, $\mathbf{Z}$, and the priors on the random variables associated with log features computed with image evidence, Z_q^f, one can marginalize over the random variables associated with visible log features and compute the maximum-likelihood assignment to the random variables associated with occluded log features, $\hat{\mathbf{Z}}^f$, that is consistent with a given structure estimate.

$$\hat{\mathbf{Z}}^f = \underset{\substack{Z_q^f \\ V_q^f = \textbf{false}}}{\operatorname{argmax}} \sum_{\substack{Z_q^f \\ V_q^f = \textbf{true} \\ \Phi[\mathbf{Z},\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w]}} \Pr(\mathbf{Z},\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w)$$

One can then ask the following question: what is the maximal amount δ that one can shift the probability mass on the random variables associated with occluded log features *away* from the uniform prior, reassigning that shifted probability mass to the opposite element of the support of that random variable from the above maximum-likelihood assignment, such that structure estimation yields the same estimated structure. Or in simpler terms,

> *How much hypothetical evidence of occluded log features is needed to cause me to change my mind away from the estimate derived from a uniform prior on such occluded features?*

We compute this δ using a modified structure estimation step

$$\underset{\mathbf{Z}}{\operatorname{argmax}} \sum_{\substack{\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w \\ \Phi[\mathbf{Z},\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v,\mathbf{Z}^w]}} \Pr(\mathbf{Z},\mathbf{Z}^+,\mathbf{Z}^-,\mathbf{Z}^u,\mathbf{Z}^v\mathbf{Z}^w) = \mathbf{Z}$$

when, for all q^f where $V_q^f = \textbf{false}$

$$\Pr(Z_q^f = \neg\hat{Z}_q^f) = \tfrac{1}{2} + \delta$$
$$\Pr(Z_q^f = \hat{Z}_q^f) = \tfrac{1}{2} - \delta$$

We call such a δ the *estimation tolerance*. Then, for any estimated structure, one can make a confidence judgment by comparing the estimation tolerance to an overall tolerance threshold δ^*. One wishes to select a value for δ^* that appropriately trades off false positives and false negatives in such confidence judgements: we want to minimize the cases that result in a positive confidence assessment for an incorrect structure estimate and also minimize the cases that result in a negative confidence assessment for a correct structure estimate. Because the methods we present in the next section can gather additional evidence in light of negative confidence assessment in structure estimation, the former are more hazardous than the latter because the former preclude

gathering such additional evidence and lead to an ultimate incorrect structure estimate while the latter simply incur the cost of such additional evidence gathering. Because of this asymmetry, our method is largely insensitive to the particular value of δ^* so long as it is sufficiently high to not yield excessive false positives. We have determined empirically that setting $\delta^* = 0.2$ yields a good tradeoff: only 3/105 false positives and 7/105 false negatives on our corpus.

One can determine the estimation tolerance by binary search for the smallest value of $\delta \in (0, 0.5)$ that results in a different estimated structure. However, this process is time consuming. But we don't actually need the value of δ; we only need to determine whether $\delta < \delta^*$. One can do this by simply asking whether the estimated structure, $\mathbf{Z}$, changes when the probabilities are shifted by δ^*

$$\Pr(Z_q^f = \neg\hat{Z}_q^f) = \tfrac{1}{2} + \delta^*$$
$$\Pr(Z_q^f = \hat{Z}_q^f) = \tfrac{1}{2} - \delta^*$$

This involves only a single new structure estimation. One can make this process even faster by initializing the branch-and-bound structure-estimation algorithm with the probability of the original structure estimate given the modified distributions for the random variables associated with occluded log features.

V. Gathering Additional Evidence to Improve Structure Estimation

Structure estimation can be made more reliable by integrating multiple sources of image evidence. We perform structure estimation in a novel robotic environment, illustrated in Fig. 2, that facilities automatically gathering multiple sources of image evidence as needed. The structures are assembled in the robot workspace. This workspace is imaged by a camera mounted on a pendulum arm that can rotate 180° about the workspace, under computer control, to image the assembly from different viewpoints. This can be used to view portions of the assembly that would otherwise be occluded. Moreover, a robotic arm can disassemble a structure on the workspace. This can be used to reveal the lower layers of a structure that would otherwise be occluded by higher layers. These methods can further be combined. Generally speaking, we seek a method for constraining a single estimate of an initial structure with multiple log features derived from different viewpoints and different stages of disassembly.

We can do this as follows. Let $\mathbf{Z}$ be a collection of random variables Z_q associated with log occupancy for a given initial structure. Given multiple views $i = 1, \ldots, n$ with collections $\mathbf{Z}_i$ of random variables Z_q^+, Z_q^-, Z_q^u, Z_q^v, and Z_q^w associated with the image evidence for log features from those views, we can compute

$$\underset{\mathbf{Z}}{\operatorname{argmax}} \sum_{\substack{\mathbf{Z}_1 \ldots \mathbf{Z}_n \\ \Phi[\mathbf{Z},\mathbf{Z}_1] \wedge \ldots \wedge [\mathbf{Z},\mathbf{Z}_n]}} \Pr(\mathbf{Z}, \mathbf{Z}_1, \ldots, \mathbf{Z}_n)$$

Only two issues arise in doing this. First, we do not know the relative camera angles of the different views. Even though one can estimate the camera-relative pose of the

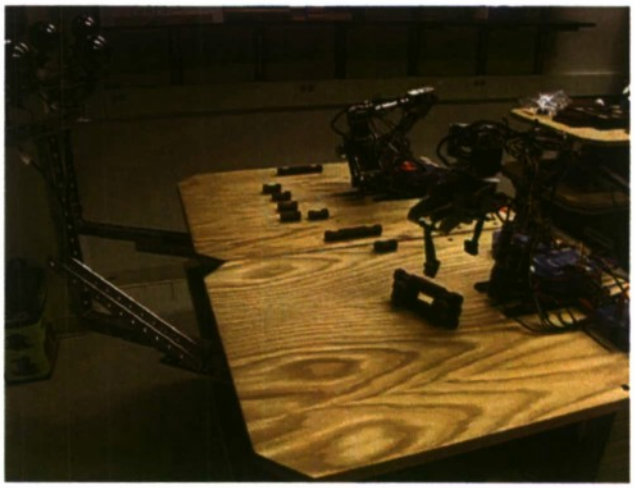


Fig. 2. Our novel robotic environment for performing structure estimation. Note that the head can rotate 180° about the workspace, under computer control, to image the assembly from different viewpoints, and the robot arm can disassemble the structure on the workspace.

structure independently for each view, this does not yield the registration between these views. There are only four possible symbolic orientations of the structure in each view so for n views we need only consider 4^{n-1} possible combinations of such symbolic orientations. We can search for the combination that yields the maximum-likelihood structure estimate. We do this search greedily, incrementally adding views to the structure-estimation process and registering each added view by searching for the best among the four possible registrations. Second, in the case of partial disassembly, we need to handle the fact that the partially disassembled structure is a proper subset of the initial structure. We do this simply by omitting random variables associated with log features for logs that are known to have been removed in the disassembly process and not instantiating constraints that mention such omitted random variables.

We can combine the techniques from section IV with these techniques to yield an active-vision [3] approach to producing a confident and correct structure estimate. One can perform structure estimation on an initial image and assess one's confidence in that estimate. If one is not confident, one can plan a new observation, entailing either a new viewpoint, a partial-disassembly operation, or a combination of the two and repeat this process until one is sufficiently confident in the estimated structure. Only one issue arises in doing this. One must plan the new observation. We do so by asking the following question:

> *Which of the available actions maximally increases confidence?*

Like before, if we could determine the conditional distribution over consistent structures given image evidence, we could compute the decrease in entropy that each available action would yield and select the action that maximally decreases entropy. But again, it is intractable to compute this distribution and further intractable to compute its entropy. Thus we adopt an alternate means of measuring increase in confidence.

Given visibility estimates V_{iq}^f for view i of the n current views along with a structure estimate $\mathbf{Z}$ constructed from those views, and priors on the random variables associated with log features computed with image evidence for each of these views Z_{iq}^f, one can marginalize over the random variables associated with visible log features, $V_{iq}^f = \textbf{true}$, and compute the maximum-likelihood assignment $\hat{\mathbf{Z}}^f$ to the random variables associated with occluded log features that is consistent with a given structure estimate:

$$\hat{\mathbf{Z}}^f = \underset{\substack{Z_{iq}^f \\ V_{iq}^f=\textbf{false}}}{\text{argmax}} \sum_{\substack{Z_{iq}^f \\ V_{iq}^f=\textbf{true} \\ \Phi[\mathbf{Z},\mathbf{Z}_1]\wedge\ldots\wedge\Phi[\mathbf{Z},\mathbf{Z}_n]}} \Pr(\mathbf{Z},\mathbf{Z}_1,\ldots,\mathbf{Z}_n)$$

We can further determine those log features that are invisible in all current views but visible in a new view j that would result from a hypothetical action under consideration. One can then ask the following question: what is the maximal amount δ' that one can shift the probability mass on these random variables *away* from the uniform prior, reassigning that shifted probability mass to the opposite element of the support of that random variable from the above maximum-likelihood assignment, such that structure estimation when adding the new view yields the same estimated structure. Or in simpler terms,

> *For a given hypothetical action, how much hypothetical evidence of log features that are occluded in all current views is needed in an imagined view resulting from that action where those log features are visible to cause me to change my mind away from the estimate derived from a uniform prior on such features?*

For an action that yields a new view, j, we compute δ' as follows

$$\underset{\mathbf{Z}}{\text{argmax}} \sum_{\substack{\mathbf{Z}_1\ldots\mathbf{Z}_n\,\mathbf{Z}_j \\ \Phi[\mathbf{Z},\mathbf{Z}_1]\wedge\ldots\wedge\Phi[\mathbf{Z},\mathbf{Z}_n]\wedge\Phi[\mathbf{Z},\mathbf{Z}_j]}} \Pr(\mathbf{Z},\mathbf{Z}_1,\ldots,\mathbf{Z}_n,\mathbf{Z}_j) = \mathbf{Z}$$

when:

$$\Pr(Z_{iq}^f = \neg\hat{Z}_{iq}^f) = \tfrac{1}{2} + \delta$$
$$\Pr(Z_{iq}^f = \hat{Z}_{iq}^f) = \tfrac{1}{2} - \delta$$

for all q^f where $V_{jq}^f = \textbf{true} \wedge (\forall i) V_{iq}^f = \textbf{false}$. Because we wish to select the action with the smallest δ', we need its actual value. Thus we perform binary search to find δ' for each hypothetical action and select the one with the lowest δ'. This nominally requires sufficiently deep binary search to compute δ' to arbitrary precision. One can make this process even faster by performing binary search on all hypothetical actions simultaneously and terminating when there is only one hypothetical action lower than the branch point. This requires that binary search be only sufficiently deep to discriminate between the available actions.

VI. Results

We gathered a corpus of 5 different images of each of 32 different structures, each from a different viewpoint, for a total of 160 images. The structures were carefully designed so that proper subset relations exist among various pairs of the 32 distinct structures.

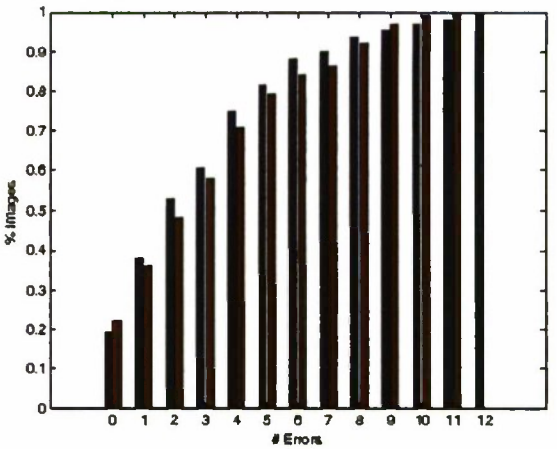


Fig. 3. Error histograms for manual visibility annotation (in blue) and automatic visibility estimation (in red). 100% of the structures estimated had 12 or fewer errors. Note that the latter performs as well as the former.

We first evaluated automatic visibility estimation. We performed combined visibility and structure estimation on 105 of the 160 images[1] and compared the maximum-likelihood structure estimate to that produced in our earlier work [19] using manual annotation of visibility. For each image, we compare the maximum-likelihood structure estimate to ground truth and compute the number of errors. We do this as follows. Each 1-, 2-, or 3-notch log in either the ground truth or estimated structure that is replaced with a different, possibly empty, collection of logs in the alternate structure counts as a single error (which may be a deletion, addition, or substitution). Further, each collection of r adjacent logs with the same medial axis in the ground truth that is replaced with a different collection of s logs in the estimated structure counts as $min(r,s)$ errors. We then compute an error histogram of the number of images with fewer than t errors. Fig. 3 shows the error histograms for manual visibility annotation and automatic visibility estimation. Note that the latter performs as well as the former. Thus our automatic visibility-estimation process appears to be reliable.

We then evaluated structure-estimation confidence assessment. We computed the false-positive rate and false-negative rate of our confidence-assessment procedure over the entire corpus of 105 images, where a false positive occurs with a positive confidence assessment for an incorrect structure estimate and a false negative occurs with negative confidence assessment for a correct structure estimate. This resulted in only 3 false positives and 7 false negatives on our corpus.

We then evaluated the active-vision process for performing actions to improve structure-estimation confidence on 90 images from our corpus.[1] So as not to render this evaluation dependent on the mechanical reliability of our robot which

[1]Due to limited computational resources we were unable to perform combined visibility, structure estimation, and active-vision on all 160 images. The final submission will contain the full set of results.

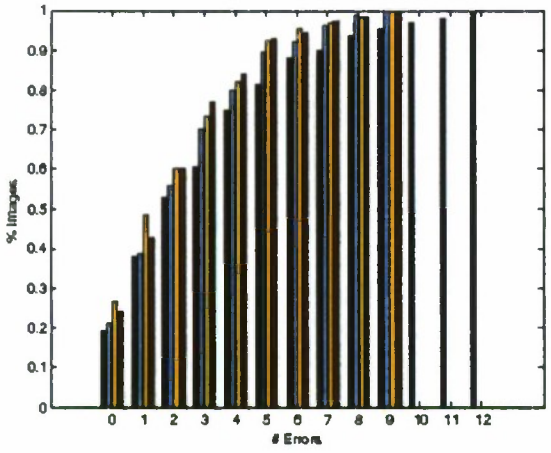


Fig. 4. Error histograms for the baseline structure estimation (in dark blue) and each of the active-vision process (partial disassembly in light blue, multiple views in yellow, and the combination of these in red).

is tangential to the current paper and focus the evaluation on the computational method, we use the fact that our corpus contains multiple views of each structure from different viewpoints to simulate moving the robot head to gather new views and the fact our corpus contains pairs of structures in a proper-subset relation to simulate using the robot to perform partial disassembly. We first evaluated simulated robot-head motion to gather new views. For each image, we took the other images of the same structure from different viewpoints as potential actions and perform our active-vision process. We next evaluated simulated robotic disassembly. For each image, we took images of proper-subset structures taken from the same viewpoint as potential actions and perform our active-vision process. We finally evaluated simulated combined robot-head motion and robotic disassembly. For each image, we took all images of proper-subset structures taken from any viewpoint as potential actions and perform our active-vision process. For each of these, we computed the error histogram at the termination of the active-vision process. Fig. 4 shows the error histograms for each of the active-vision processes together with the error histogram for baseline structure estimation from a single view on this subset of 90 images. Fig. 5 shows a rendering of the final estimated structure when performing each of the four processes from Fig. 4 on the same initial image. Log color indicates correct (green) or incorrect (red) estimation of log occupancies. Note that our active-vision processes consistently reduce estimation error.

VII. Related Work

Our work shares three overall objectives with prior work:

- estimating 3D structure from 2D images,
- determining when there is occlusion, and
- active vision.

However, our work explores each of these issues from a novel perspective.

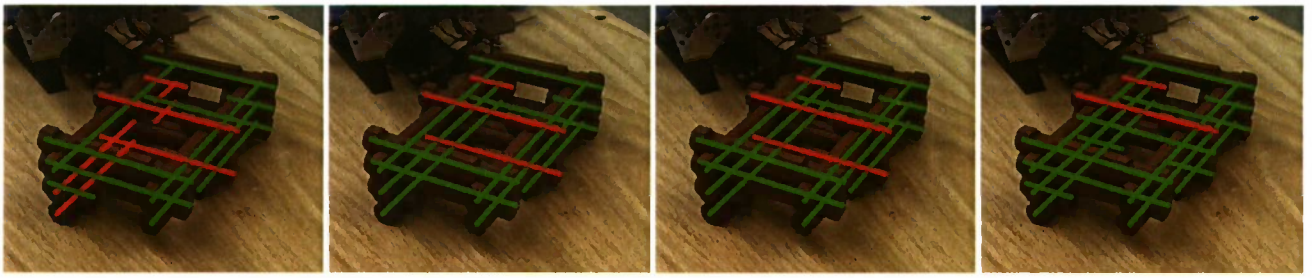
Prior work on structure estimation (e.g. [8], [12], [17]) focuses on *surface estimation*, recovering a 3D surface from 2D images. In contrast, our work focuses on recovering the *constituent structure* of an assembly: what parts are used to make the assembly and how such parts are combined. Existing state-of-the-art surface reconstruction methods (e.g. Make3D [16]) are unable to determine surface structure of the kinds of LINCOLN LOG assemblies considered here. Ever if such surface estimates were successful, such estimates alone are insufficient to determine the constituent structure.

Prior work on occlusion determination (e.g. [8], [9]) focuses on finding occlusion boundaries: the 2D image boundaries of occluded regions. In contrast, our work focuses on determining occluded *parts* in the constituent structure. We see no easy way to determine occluded parts from occlusion boundaries because such boundaries alone are insufficient to determine even the number of occluded parts, let alone their types and positions in a 3D structure.

Prior work on active vision (e.g. [15]) focuses on integrating multiple views into surface estimation and selecting new viewpoints to facilitate such in the presence of occlusion. In contrast, our work focuses on determining the confidence of constituent structure estimates and choosing an action with maximal anticipated increase in confidence. We consider not only viewpoint changes but also robotic disassembly to view object interiors. Also note that the confidence estimates used in our approach to active vision are mediated by the visual language model. We might not need to perform active vision to observe all occluded structure as it might be possible to infer part of the occluded structure. Prior work selects a new viewpoint to render occluded structure visible. We instead select an action to maximally increase confidence. Such an action might actually not attempt to view an occluded portion of the structure but rather increase confidence in a visible portion of the structure in a way that when mediated by the language model ultimately yields a maximal increase in the confidence assessment of a portion of the structure that remains occluded even with the action taken.

VIII. Conclusion

We have presented a general framework for (a) seeing the unseeable, (b) seeing unseeability, (c) a rational basis for determining confidence in what one sees, and (d) an active-vision decision-making process for determining rational behavior in the presence of unseeability. We instantiated and evaluated our general framework in the LINCOLN LOG domain and found it to be effective. This framework has many potential extensions. One can construct random variables to represent uncertain evidence in other modalities such as language and speech and one can augment the stochastic CSP to mutually constraint these variables together with the current random variables that represent image evidence and latent structure so that a latent utterance describes a latent structure. One can then use the same maximum-likelihood estimation techniques to produce the maximum-likelihood utterance consistent with a structure marginalizing over image evidence. This constitutes producing an utterance that describes a visual observation. One can use the same maximum-likelihood estimation techniques to produce the



(a) (b) (c) (d)

Fig. 5. Rendered structure for the following four methods: (a) Baseline structure estimation. (b) Partial disassembly. (c) Multiple views. (d) Combined partial disassembly and multiple views.

maximum-likelihood sequence of robotic actions consistent with building a structure marginalizing over utterance evidence or alternatively image evidence. This would constitute building a structure by understanding a linguistic description of that structure or by copying a visually observed assembly. One can combine evidence from an uncertain visual perception of a structure with evidence from an uncertain linguistic description of that structure to reduce the uncertainty of structure estimation. This would constitute using vision and language to mutually disambiguate each other. One could augment one's collection of potential actions to include speech acts as well as robotic-manipulation actions and search for the action that best improves confidence. This would constitute choosing between asking someone to provide you information and seeking that information yourself. One could determine what another agent can see from what that agent says. Likewise one could decide what to say so that another agent can see what is unseeable to that agent yet is seeable to you. Overall, this can lead to a rational basis for cooperative agent behavior and a theory of the perception-cognition-action loop which incorporates mutual belief, goals, and desires where agents seek to assist each other by seeing what their peers cannot, describing such sight, and inferring what their peers can and cannot see. We are currently beginning to investigate these potential extensions to our general approach and hope to present them in the future.

ACKNOWLEDGMENTS

This work was supported, in part, by NSF grant CCF-0438806, by the Naval Research Laboratory under Contract Number N00173-10-1-G023, by the Army Research Laboratory accomplished under Cooperative Agreement Number W911NF-10-2-0060, and by computational resources provided by Information Technology at Purdue through its Rosen Center for Advanced Computing. Any views, opinions, findings, conclusions, or recommendations contained or expressed in this document or material are those of the author(s) and do not necessarily reflect or represent the views or official policies, either expressed or implied, of NSF, the Naval Research Laboratory, the Office of Naval Research, the Army Research Laboratory, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

REFERENCES

[1] R. Baillargeon. Representing the existence and the location of hidden objects: Object permanence in 6- and 8-month-old infants. *Cognition*, 23:21–41, 1986.

[2] R. Baillargeon. Object permanence in $3\frac{1}{2}$- and $4\frac{1}{2}$-month-old infants. *Developmental Psychology*, 23(5):655–64, 1987.

[3] R. Bajcsy. Active perception. In *Proceedings of IEEE*, volume 76, pages 966–1005, Aug. 1988.

[4] L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3:1–8, 1972.

[5] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–71, 1970.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[7] J. J. Freyd, T. M. Pantzer, and J. L. Cheng. Representing statics as forces in equilibrium. *Journal of Experimental Psychology, General*, 117(4):395–407, Dec. 1988.

[8] A. Gupta, A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision*, volume 6314, pages 482–496, 2010.

[9] D. Hoiem, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *International Journal of Computer Vision*, 91(3):328–346, 2011.

[10] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[11] J.-L. Lauriere. A language and a program for stating and solving combinatorial problems. *Artificial Intelligence*, 10(1):29–127, 1978.

[12] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, June 2009.

[13] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[14] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[15] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15:417–433, May 1993.

[16] A. Saxena, M. Sun, and A. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(5):824–840, may 2008.

[17] A. Saxena, M. Sun, and A. Y. Ng. Learning 3-d scene structure from a single still image. In *ICCV workshop on 3D Representation for Recognition 3dRR07*, 2007.

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[19] N. Siddharth, A. Barbu, and J. M. Siskind. A visual language model for estimating object pose and structure in a generative visual domain. In *Proceedings of the IEEE International Conf. on Robotics and Automation*, May 2011.

[20] A. Streri and E. S. Spelke. Haptic perception of objects in infancy. *Cognitive Psychology*, 20(1):1–23, 1988.

[21] K. Wynn. Psychological foundations of number: Numerical competence in human infants. *Trends in Cognitive Sciences*, 2:296–303, 1998.